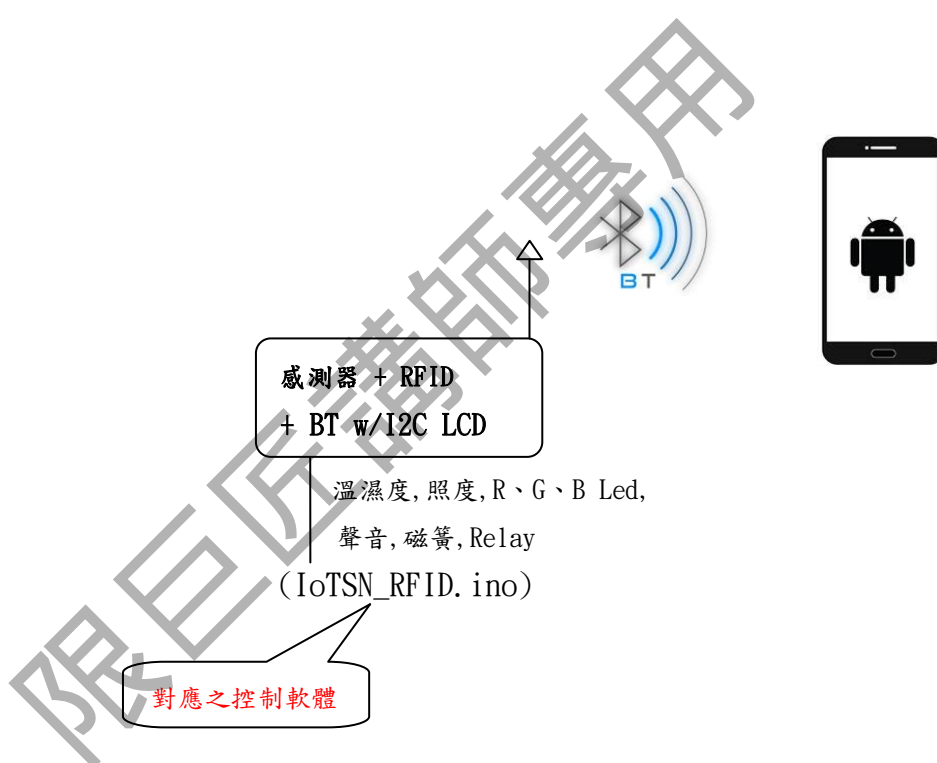


# 第 1 堂：物聯網趨勢與物聯網通訊介紹

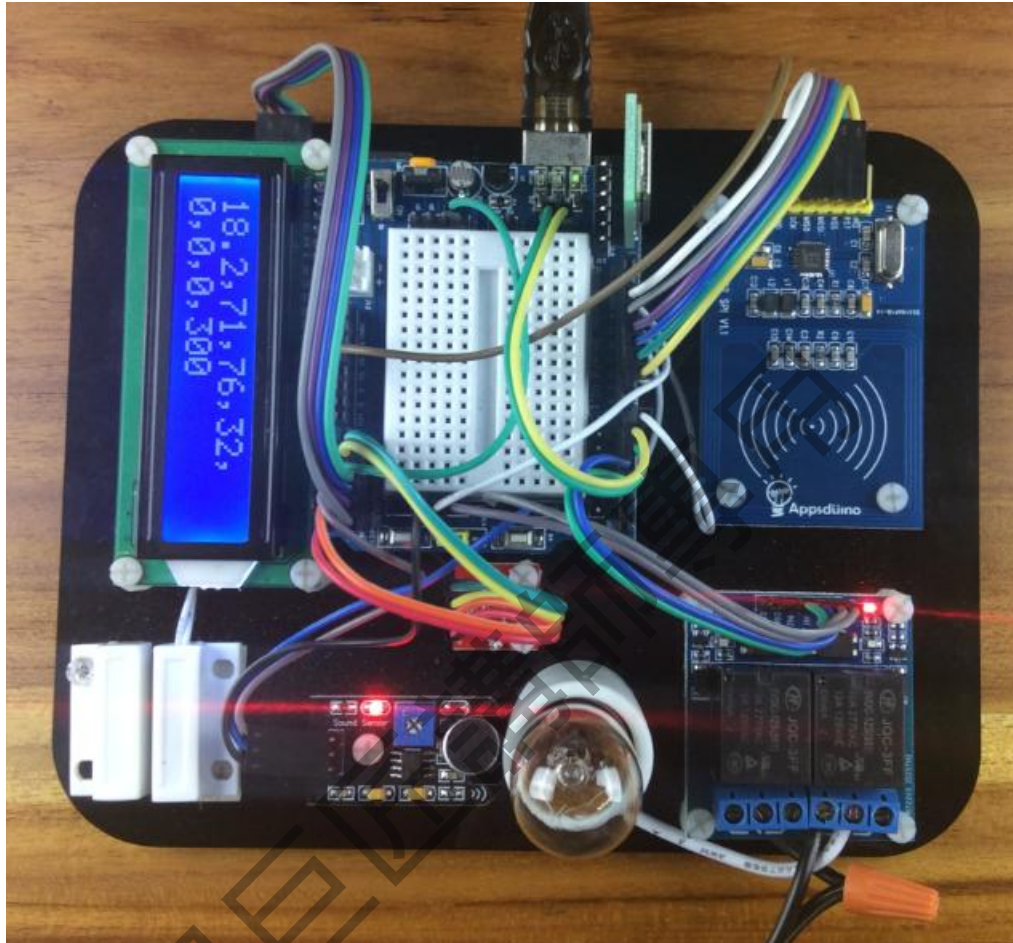
## 智慧家庭開發套件組裝與介紹

本套件物聯網智慧家庭學習系統，包含 RFID 與感測器以及手持行動裝置 APP 監控畫面。感測器配置有溫濕度、照度、聲音、開窗等感測器，以及可從遠端控制的燈光開關。RFID Reader 與 Tag 可做為門禁進出管理，而控制板上的感測器及開關最新狀況，除了將其顯示在 LCD 螢幕外，資訊也會即時透過藍牙(Bluetooth)無線模組，轉送給 Android 手持裝置內建的 APP 監測程式，因此使用者可在無線的情境下，由手持裝置觀看目前即時的溫濕度、照度、聲音，並由遠端控制開關啟動(或關閉)，其系統示意圖如下：



物聯網(IoT)感測應用發展系統示意圖

而組裝後之套件如下圖示：



物聯網(IoT)感測應用發展系統組裝圖



## IoTSN 互動 App

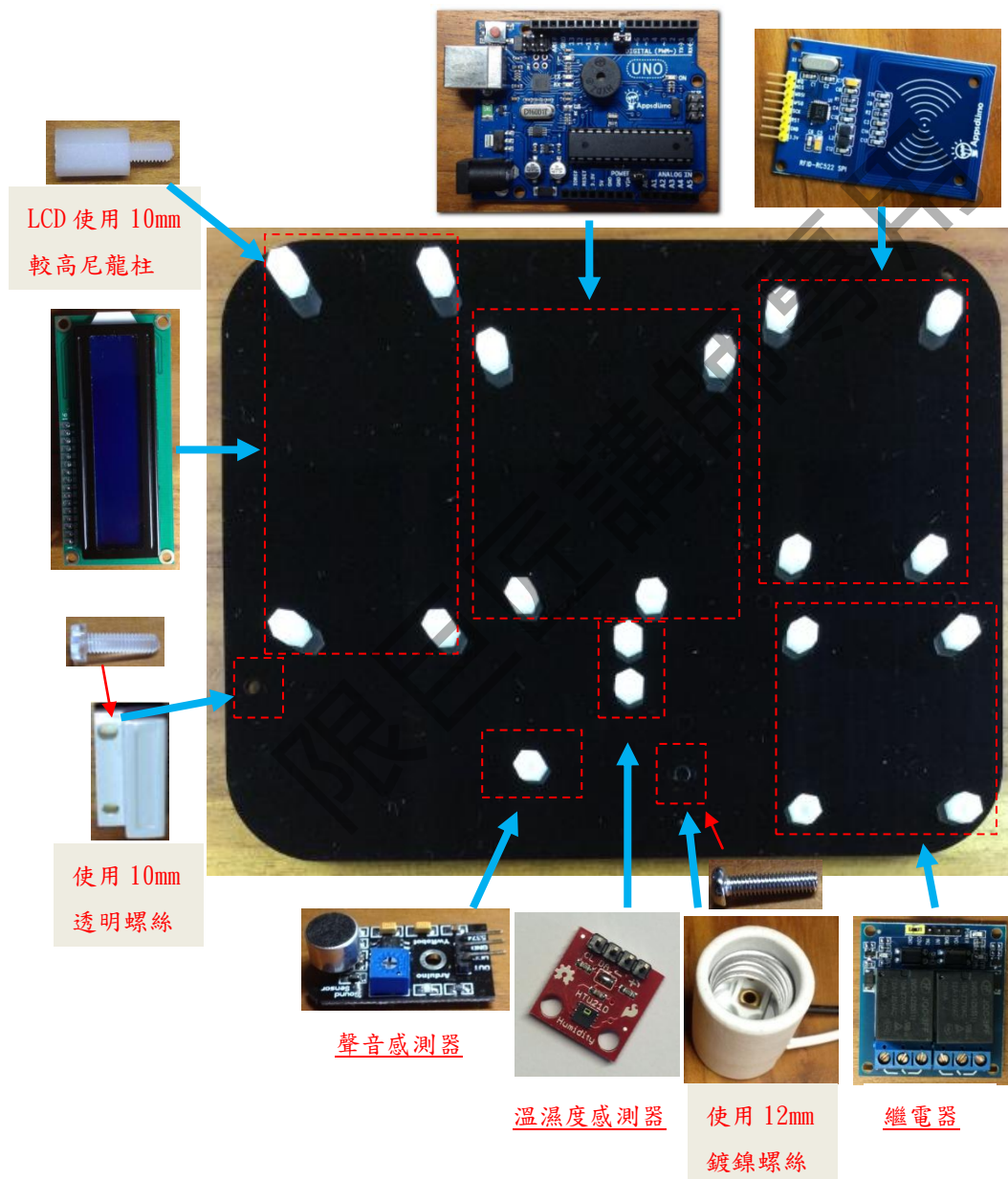


## IoTSN 互動 App Google Play 下載

## 軟硬體組裝

### I. 安裝板子與感測器

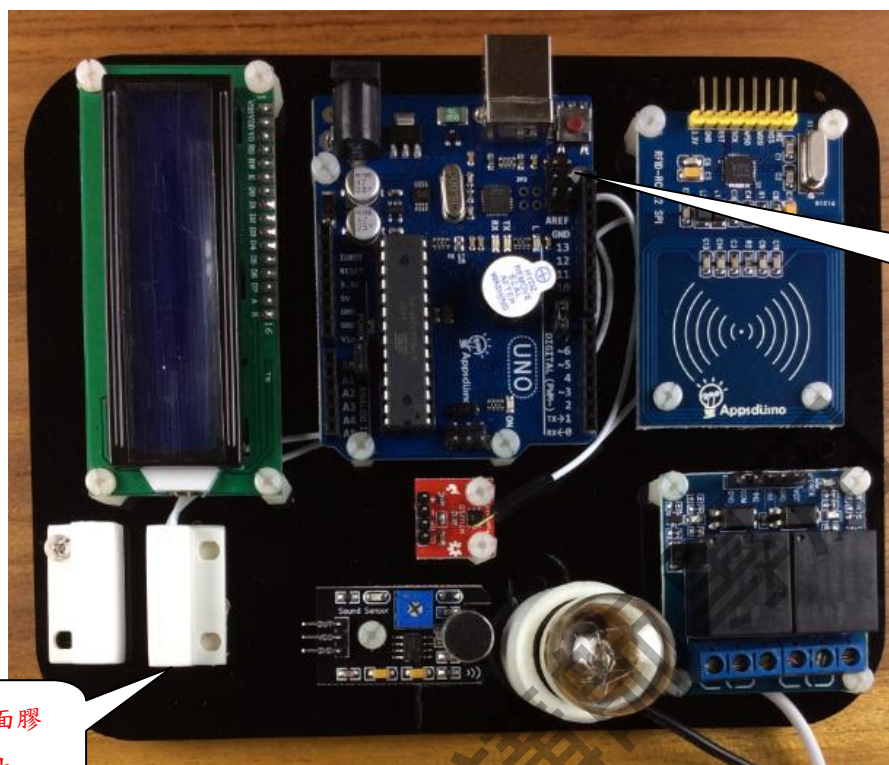
1. 將底板之保護貼紙移除並鎖上六角尼龍柱(先不鎖緊，待裝上感測器或模組後再鎖緊)，相對位置如圖下：



組裝位置示意圖

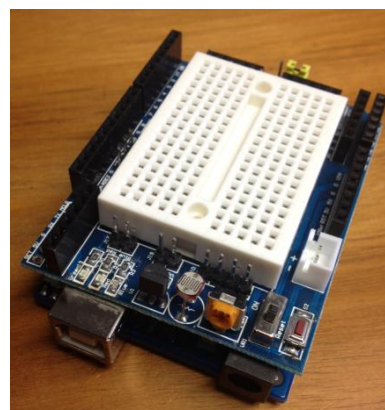


2. 將各模組與感測器鎖上螺絲固定在底板(如下圖)：



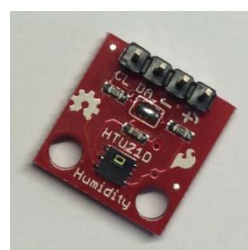
3. 安裝 Appsdüino 擴充版

將實驗控制板依下列順序堆疊  
上層：Appsdüino 擴充實驗板 + 麵包版  
下層：Appsdüino UNO 控制板



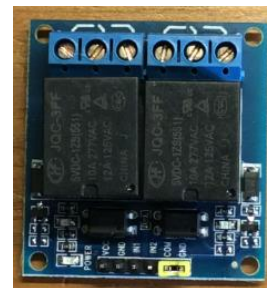
4. 安裝溫濕度感測器

接腳	名稱	Ardino 腳位
1	CL	串列時鐘線，接A5
2	DA	串列資料線，接A4
3	-	接GND
4	+	3.3V



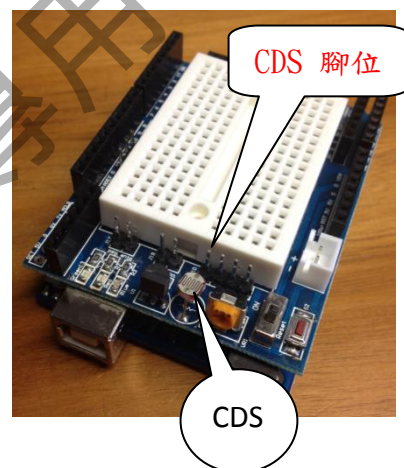
## 5. 安裝繼電器模組

Pin	ID	繼電器模組
1	VCC	接 5V
2	GND	接 GND
3	IN1	Arduino D4
4	IN2	Arduino D3 Pin



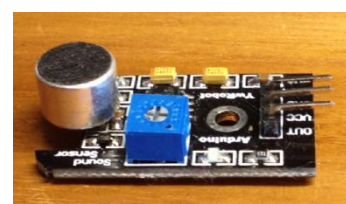
## 6. 安裝照度計

Pin	ID	照度計(CDS)
1	CDS	接類比 A2 Pin



## 8. 安裝聲音感測器

Pin	ID	聲音感測器
1	OUT	類比輸出接類比 A3 Pin
2	VCC	接 5V
3	GND	接 GND



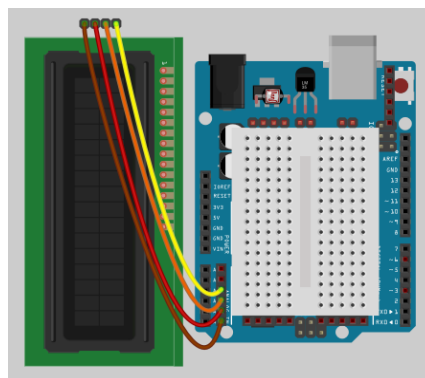
## 9. 磁簧開關

Pin	ID	磁簧感測器
1	一端	接 Arduino D7
2	另一端	接 GND



## 10. 安裝 I2C LCD 顯示器(I2C 位址 : 0x27)

Pin	ID	說明
1	GND	接 GND
2	VCC	接 5V
3	SDA	Data 接 Arduino 類比 A4 Pin
4	SCL	Clock 接 Arduino 類比 A5 Pin



## 11. 安裝 RFID 讀寫模組(採用 SPI 介面)

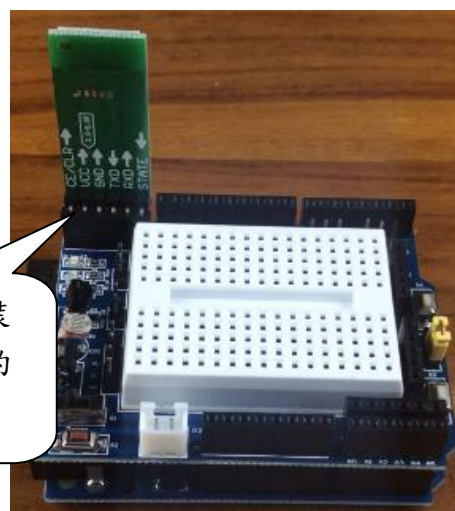
接腳	名稱	RFID Reader(SPI 介面)
1	IRQ	X
2	RST	接 Arduino D9
3	NSS	接 Arduino D10
4	MOSI	接 Arduino D11
5	MISO	接 Arduino D12
6	SCK	接 Arduino D13
7	GND	接 GND
8	3.3V	接 3.3V



## 12. 安裝無線藍牙模組

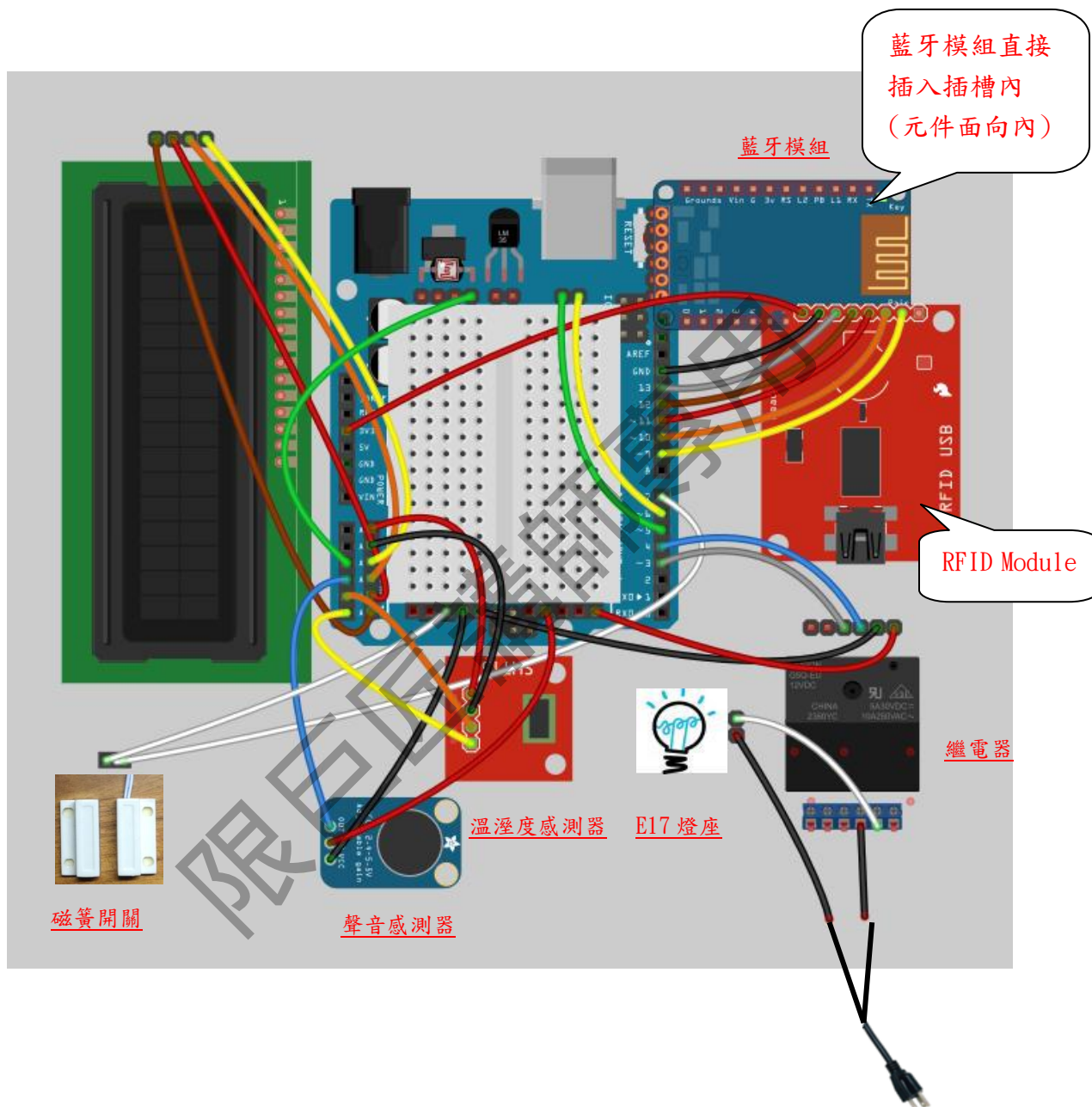


藍牙模組安裝  
在控制板上的  
插座上

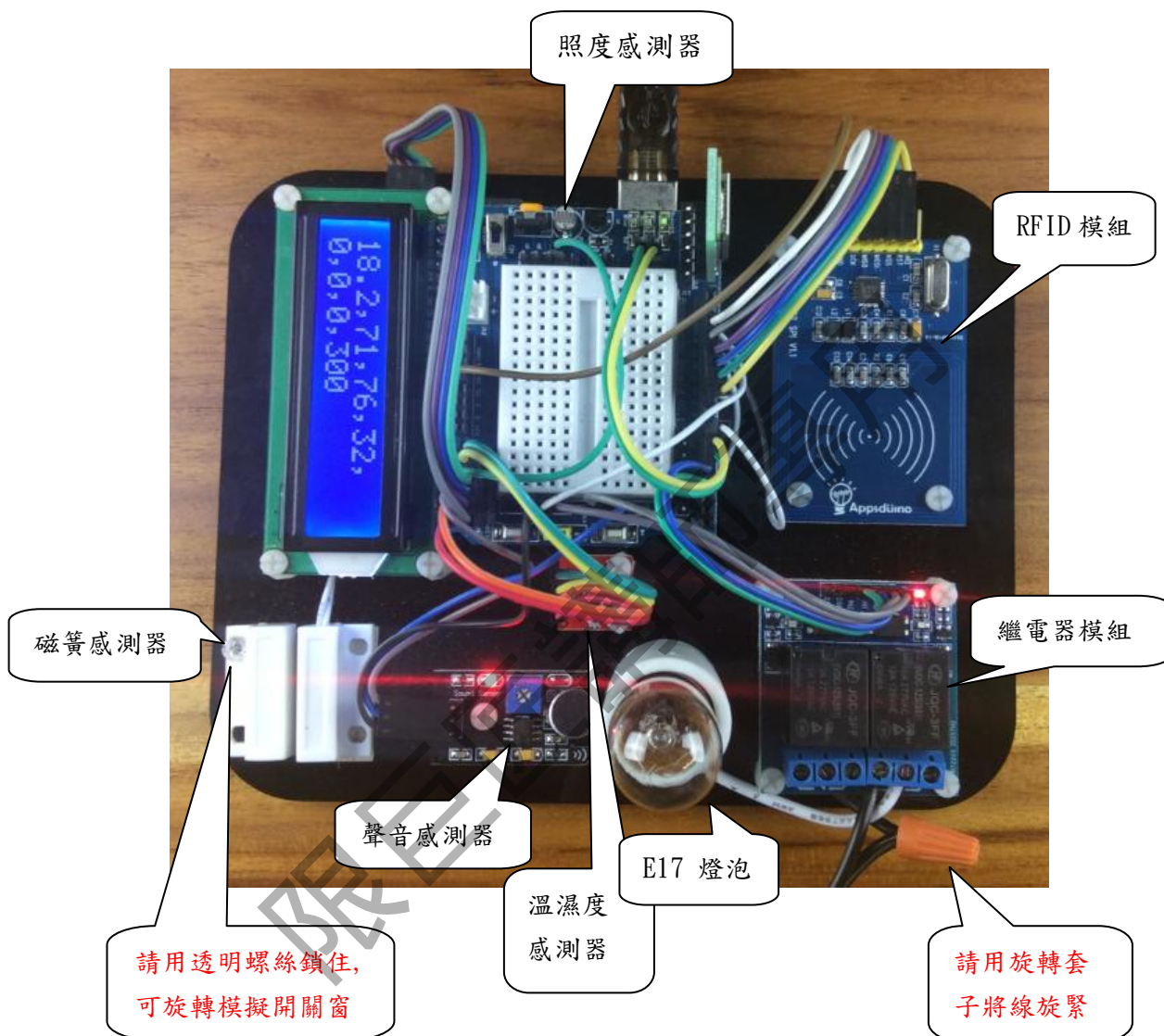




◆ 感測系統 I/O 對應接腳接線圖(此乃模擬 UNO 接上 Appsdüino 擴充板之接線圖)







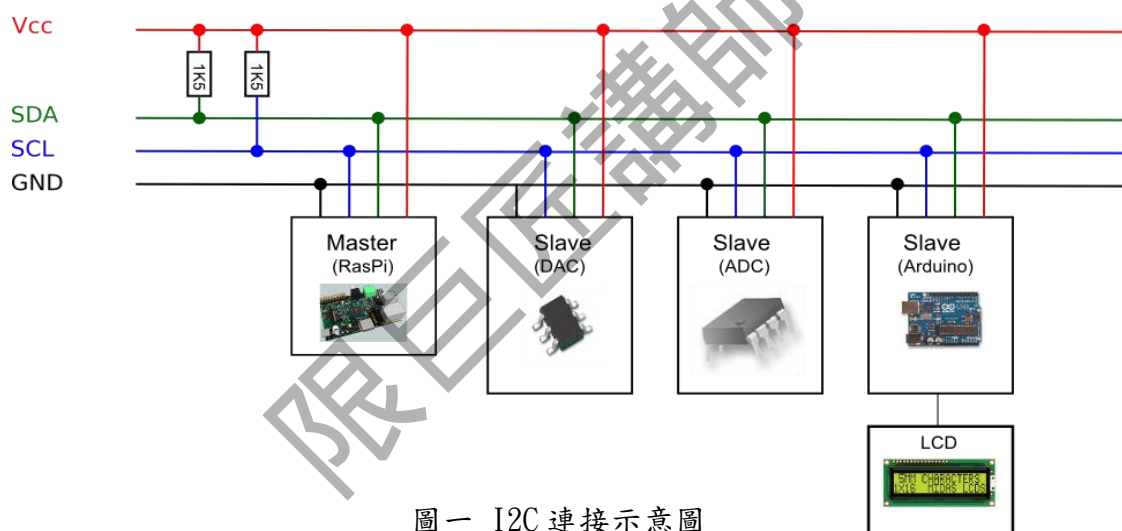
組裝完畢後的圖片

## 第 2 堂：智慧家庭套件感測器介紹

### IIC/I2C Bus 簡介

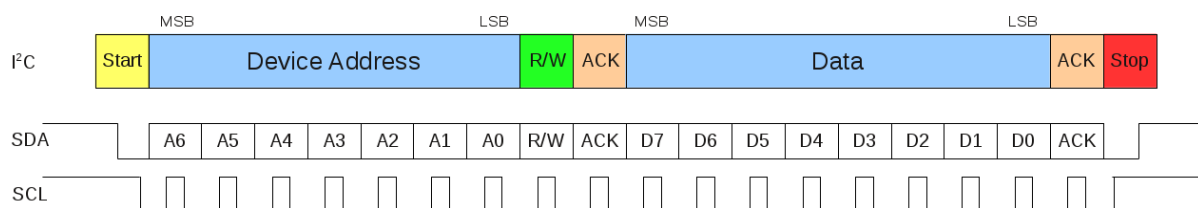
I2C Bus 是由菲利浦公司(Philips Semiconductors)在 80 年代時就開發出來的技術，其目的為提供一個簡單的方式來連接 CPU 與週邊晶片。在嵌入式系統中，MCU 通常以 Memory-Map I/O 的方式來定址週邊裝置，並使用微控制器的並列位址線與資料線來做連接。這種方式造成了 PCB 電路板中有大量的位址線與資料線，而且還要加上位址解碼器與附加的邏輯電路。這對於大量生產或需求輕薄短小的產品是無法接受的。若使用 I2C Bus 來做連接，則能省下大量的元件，製造商則能有效的增加其產品的效率，且產品的價格也更能被消費者所接受。

I2C bus 為 Inter-IC bus 之縮寫，如同字面上所表達的意思，其功用為提供 IC 之間的連接與通訊之用。而 I2C Bus 之所以能克服上述問題，其原因在於 I2C Bus 只有兩條線，即序列資料線(SDA)與時脈訊號(SCL)。其中 SDA 為 Serial Data line，SCL 為 Serial Clock line，兩條線皆可做雙向傳輸。其連接示意圖如圖一所示。



圖一 I2C 連接示意圖

如圖一所示，I2C 介面是由資料線(SDA)及時脈訊號(SCL)所組成的序列介面，每一個 接到序列介面的 IC 都有自己唯一的位址，透過啟動、位址或資料傳輸、回應及結束四種主要的 模式進行資料的溝通，如圖二所示。

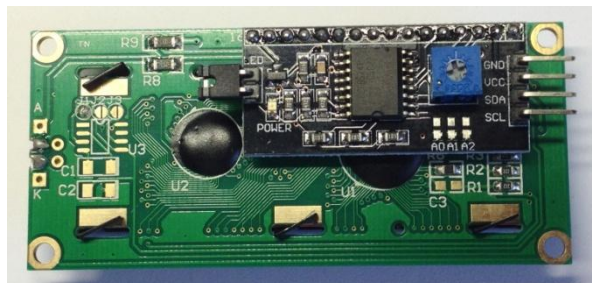


圖二 I2C 訊號示意圖

## IIC/I2C 1602 LCD 溫度顯示實驗

### ◆ LCD 模組簡介

Arduino 控制板 IO 腳位只有 20 個，加些感測器或其他元件後，IO 就不夠用了。若要接個 1602 液晶顯示器則需要 7 個 IO 才能完成，但若透過 I2C 則只需 2 個 IO 即可完成。



### ◆ 模組特性

- 電壓：+5V
- 支援 I2C 協定
- 具有背光 Led 和對比度調節電位器
- 4 線輸出
- I2C 位址：0x27

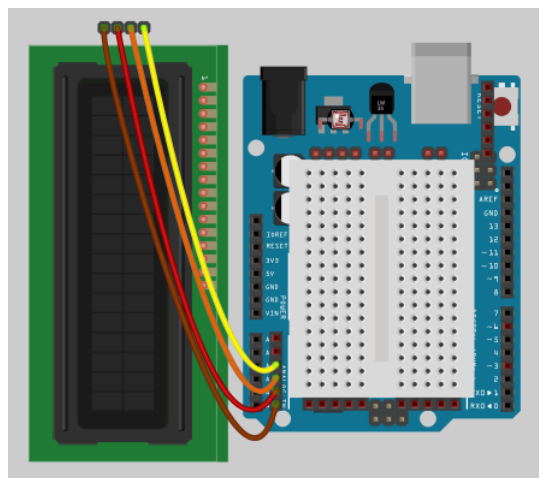
### ◆ 接腳說明(選擇使用 I2C/IIC 介面)

其腳位定義如下：

Pin	ID	說明
1	GND	接 GND
2	VCC	接 5V
3	SDA	Data 接 Arduino 類比 A4 Pin
4	SCL	Clock 接 Arduino 類比 A5 Pin

### ◆ 軟體常用函數使用說明(LCD Library)

- `begin(cols, rows)`：設定 LCD 的行與列的數目
- `clear()`：清除螢幕並將游標移至左上角
- `setCursor(col, row)`：將游標移至(col, row)位置
- `backlight()`：打開背光 Led
- `noBacklight()`：關掉背光 Led
- `print(val, format)`：將 val 顯示在 Lcd 上
- `scrollDisplayLeft()`：向左循環顯示
- `scrollDisplayRight()`：向右循環顯示



## I2C 1602 LCD 程式範例：

請將所附之 **libraries** 拷貝至安裝 **arduino IDE** 目錄之 **libraries** 資料夾內(若有相同，取代之)

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h> // F Malpartida's NewLiquidCrystal library
//需下載 LiquidCrystal_I2C Library : https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
//並將其取代原先 LiquidCrystal library, 可將其移至他處或改名即可

#define I2C_ADDR    0x27 // Define I2C Address for the PCF8574T
//---(Following are the PCF8574 pin assignments to LCD connections )---
#define BACKLIGHT_PIN 3
#define LED_OFF 1
#define LED_ON 0

/*-----( 宣告 I2C LCD 物件/Declare objects )-----*/
LiquidCrystal_I2C lcd(I2C_ADDR, 2, 1, 0, 4, 5, 6, 7); // declare I2C LCD object

void setup() /*-----( SETUP: RUNS ONCE )-----*/
{
    lcd.begin(16,2); // initialize the lcd
    // Switch on the backlight
    lcd.setBacklightPin(BACKLIGHT_PIN, POSITIVE);
    lcd.setBacklight(LED_ON); // Turn on 背光 LED
    lcd.backlight(); //Backlight ON if under program control
    lcd.setCursor(0,0); //Start at character 0 on line 0
    lcd.print("Hello, world!"); //從位置第 0 行起頭(0,0)開始顯示
    lcd.setCursor(0,1); // 設定游標位置在第一行起頭
    lcd.print("Appsduino");
} // END Setup

void loop()
{
} // END Loop
```



## I2C 1602 LCD 顯示溫度(DS18B20)程式範例：

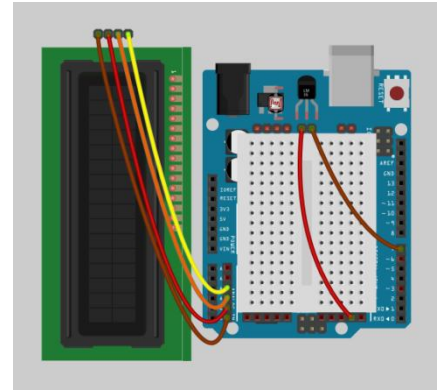
```
//-----
// 每秒讀一次溫度(DS18B20)並顯示在 LCD 上
//-----

#include <Streaming.h>
/*-----( Import needed libraries )-----*/
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h> // F Malpartida's NewLiquidCrystal library
//-----( Declare Constants )-----
#define I2C_ADDR    0x27 // Define I2C Address for the PCF8574T
//---(Following are the PCF8574 pin assignments to LCD connections )---
#define BACKLIGHT_PIN  3
#define LED_OFF  1
#define LED_ON   0
/*-----( Declare objects )-----*/
LiquidCrystal_I2C  lcd(I2C_ADDR, 2, 1, 0, 4, 5, 6, 7); // declare I2C LCD object
//----- DS18B20 Temperature sensor -----
#define DS18B20_Pin  7 //Define DS18S20 onewire signal pin on D7
#include <OneWire.h>
#include <DS18B20.h>
DS18B20 dd(DS18B20_Pin); // on digital pin 7
//-----

void setup()
{
    lcd.begin(16, 2); //Initialize LCD as 16 x 2
    // Switch on the backlight
    lcd.setBacklightPin(BACKLIGHT_PIN, POSITIVE);
    lcd.setBacklight(LED_ON);
    lcd.backlight(); //Backlight ON if under program control
    lcd.print("Temp : " );
}

void loop()
{
    float Temp ;
    Temp = dd.getTemperature(); // read the temperature from DS18B20
    lcd.setCursor(7,0); // Display from position 7 and line 0
    lcd << _FLOAT(Temp, 2) << (char)(0xDF) << "C " ;// LCD degree "o" Char
    delay(1000) ;
}
```

請將所附之 **libraries** 拷貝至安裝 **arduino IDE** 目錄之 **libraries** 資料夾內(若有相同，取代之)



## 繼電器介紹與燈光控制實驗

### ◆ 繼電器模組簡介

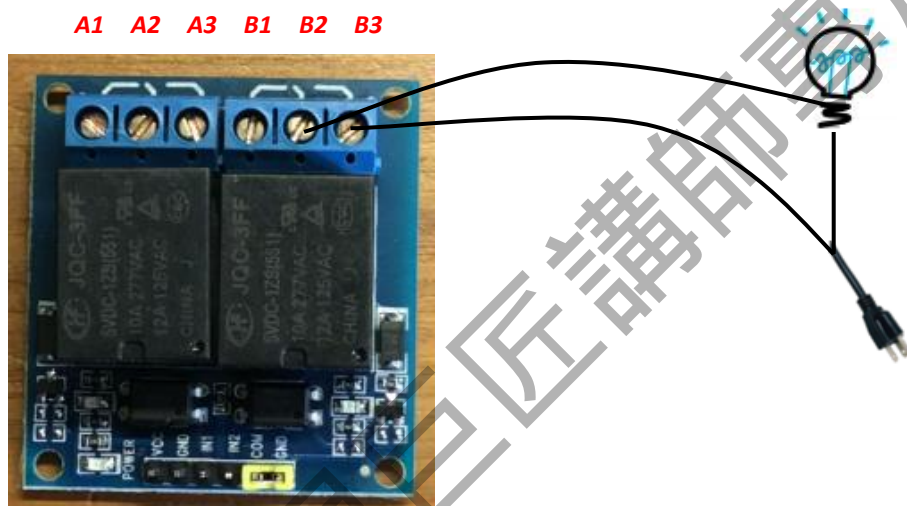
可用單晶片的 5V I/O 腳位去控制 AC 110/240V 的電氣設備，此模組具有 2CH 輸出控制能力，採用光電隔離方式抗干擾能力強，且備有二極體保護，模組下方有四個接點，與 Arduino 實驗板接線方式如下所述，目前接線使用常開(NO/Normal Open)方式，當 IN1/IN2 高電位時則繼電器閉合導通，因此可用來控制開關：

VCC--接到 Arduino 擴充板的 5V

GND--接到 Arduino 擴充板的 GND

IN1—連接 Arduino 擴充板的 Digital Pin 4(可根據需要修改不同腳位)

IN2—連接 Arduino 擴充板的 Digital Pin 3



當 Pin 4 送出 0V 低電位到 IN1，左邊繼電器 A2 A3 不導通

當 Pin 4 送出 5V 高電位到 IN1，左邊繼電器 A2 A3 導通

當 Pin 3 送出 0V 低電位到 IN2，右邊繼電器 B2 B3 不導通

當 Pin 3 送出 5V 高電位到 IN2，右邊繼電器 B2 B3 導通

### ◆ 接腳說明

Pin	ID	Arduino 腳位
1	VCC	接 5V
2	GND	接 GND
3	IN1	接數位腳位(例：D4 Pin)
4	IN2	接數位腳位(例：D3 Pin)

### ◆ 繼電器測試範例程式

可藉由 Arduino pin 4、pin 3 控制繼電器動作，用來控制所連接的家電用品啟動。  
參考範例程式如下，每隔 1 秒鐘開啟/關閉繼電器一次

```
#define Relay_IN1 4 // define Pin #
#define Relay_IN2 3

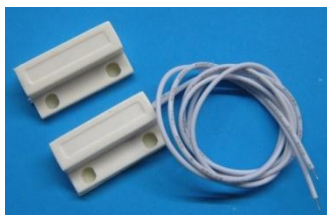
void setup() {
  pinMode(Relay_IN1, OUTPUT);
  pinMode(Relay_IN2, OUTPUT);
}

void loop() {
  digitalWrite(Relay_IN1, HIGH);
  digitalWrite(Relay_IN2, HIGH);
  delay(1000);
  digitalWrite(Relay_IN1, LOW);
  digitalWrite(Relay_IN2, LOW);
  delay(1000);
}
```

限巨匠講師專用

## 磁簧開關介紹與實驗

### ◆ 磁簧開關模組簡介



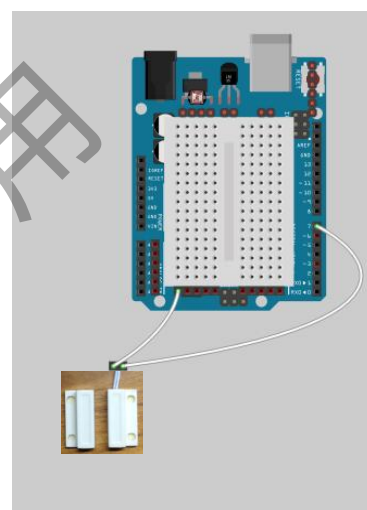
磁簧開關為一磁感應元件，常為某種設備欲定位、計次、了解設備目前狀態而設置，也廣泛使用於安全監控上，一般由一方磁鐵裝置；另一方為磁簧感應器搭配，可垂直方向感應或水平方向感應。

### ◆ 特性

- 動作情形為非接觸性，完全隔離，安全性高。
- 磁簧元件輸入阻抗相當高，可高達 1000M 歐姆以上，抗干擾強

### ◆ 接腳說明

Wire	ID	Arduino 腳位
1	GND	一端接 GND
2	DI	另一端接數位輸入 I/O(例如: D7 但啟動上拉電阻)



### ◆ 程式控制說明

```
int sensorPin = 7 ;    // select the input pin for the sensor
int sensorValue ;

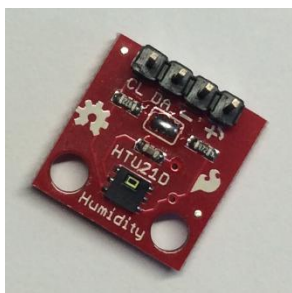
void setup() {
  pinMode(sensorPin, INPUT_PULLUP) ; // enable pull up resister
  Serial.begin(9600);
  Serial.println("Starting...");
}

void loop() {
  sensorValue = digitalRead(sensorPin);
  if(sensorValue == LOW) {
    Serial.println(".. Close detected ..") ;
  }
  else {
    Serial.println(".. Open Detected ..");
  }
  Serial.println(sensorValue);
  delay(100);
}
```

當磁鐵裝置貼近時則  
導通(低電位/LOW)



## 溫溼度感測器介紹與偵測實驗



### ◆ I2C 溫濕度模組簡介

HTU21D 超小體積的 I2C 溫濕度數位輸出模組，是基於法國 Humirel 公司高性能的溫濕度感應元件 HTU21D 設計而成。HTU21D 溫濕度感測器的尺寸僅為長寬 3x3mm(LxW)，高度 1.1mm，提供標準的 I2C 數位輸出格式、寬的工作電壓範圍、極低的功耗，以及具有很好的溫濕度精度範圍。因此 HTU21D 溫濕度數位輸出模組是具有良好的品質、快的回應速度、抗干擾能力強且性能優良的產品。

HTU21D 同時可以和瑞士的 SHT20、SHT21 實現完全相容，硬體接腳上可以實現 Pin to Pin，無需修改電路板。軟體程式方面也和瑞士的 SHT20、SHT21 相同，可以實現相容替換，無需做出任何修改。

### ◆ 感測器重要參數：

供電電壓：1.5V—3.6V

濕度測量範圍：0—100%RH

溫度測量範圍：-40°C ~ +125°C

溫度精度範圍（5°C ~ +60°C）：± 0.3°C

最大消耗功率：2.7uW

通信方式：I2C

濕度精度範圍（10%RH to 95%RH）：± 2%RH

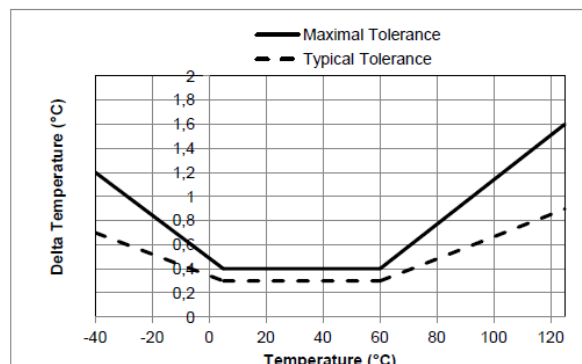
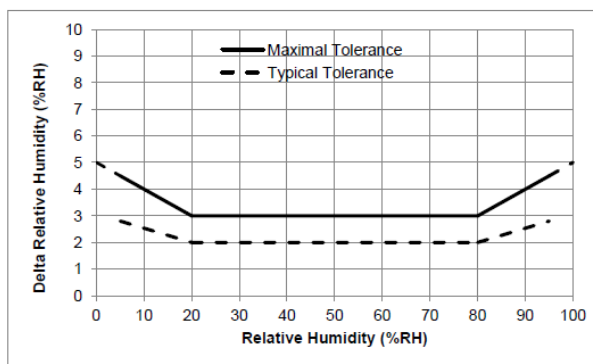
濕度遲滯：±1%RH

測量時間：50ms

年漂移量：-0.5%RH/year

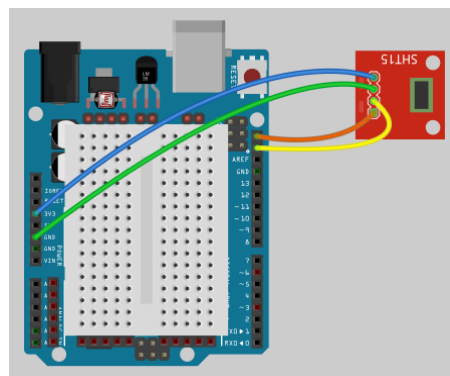
回應時間：5s

### ◆ 性能



## ◆ 接腳說明

接腳	名稱	Ardino 腳位
1	CL	串列時鐘線，接A5
2	DA	串列資料線，接A4
3	-	接GND
4	+	3.3V



- 電源引腳 (+ -)  
HTU21D 的供電電壓範圍為1.5V ~ 3.6V，因此請接3.3V。
- 串列時鐘輸入 (SCL)  
SCL/CL 引腳用於感測器I2C通訊時鐘線用途，即用於微處理器與HTU21D之間的通訊同步
- 串列資料 (SDA)  
SDA/DA 引腳為三態結構，用於讀、寫感測器資料。

## ◆ 程式控制說明

- 將 libraries 子目錄 HTU21D 內容，複製到 arduino\libraries\\*. \* 之內
- HTU21D 感測器的 I2C 的地址 (SLAVE ADDRESS) 為 0x40  
HTU21D 感測器每個 I2C 位址都相同，且為 0x40。因此在同一條線上只能連接一個 HTU21D 感測器，感測器只有在收到起始信號及與本身 I2C 位址相同時才会有回應。

測試程式如下

```
/*
HTU21D Humidity Sensor Example Code
By: Nathan Seidle
SparkFun Electronics
Date: September 15th, 2013
License: This code is public domain but you buy me a beer if you use this and we meet
someday (Beerware license).

Uses the HTU21D library to display the current humidity and temperature

Open serial monitor at 9600 baud to see readings. Errors 998 if not sensor is detected.
Error 999 if CRC is bad.

Hardware Connections (Breakoutboard to Arduino):
-VCC = 3.3V
-GND = GND
-SDA = A4 (use inline 330 ohm resistor if your board is 5V)
-SCL = A5 (use inline 330 ohm resistor if your board is 5V)

*/

#include <Wire.h>
#include <HTU21D.h>

//Create an instance of the object
HTU21D myHumidity;

void setup()
{
  Serial.begin(9600);
  Serial.println("HTU21D Example!");

  myHumidity.begin();
}

void loop()
{
  float humd = myHumidity.readHumidity();
  float temp = myHumidity.readTemperature();
```

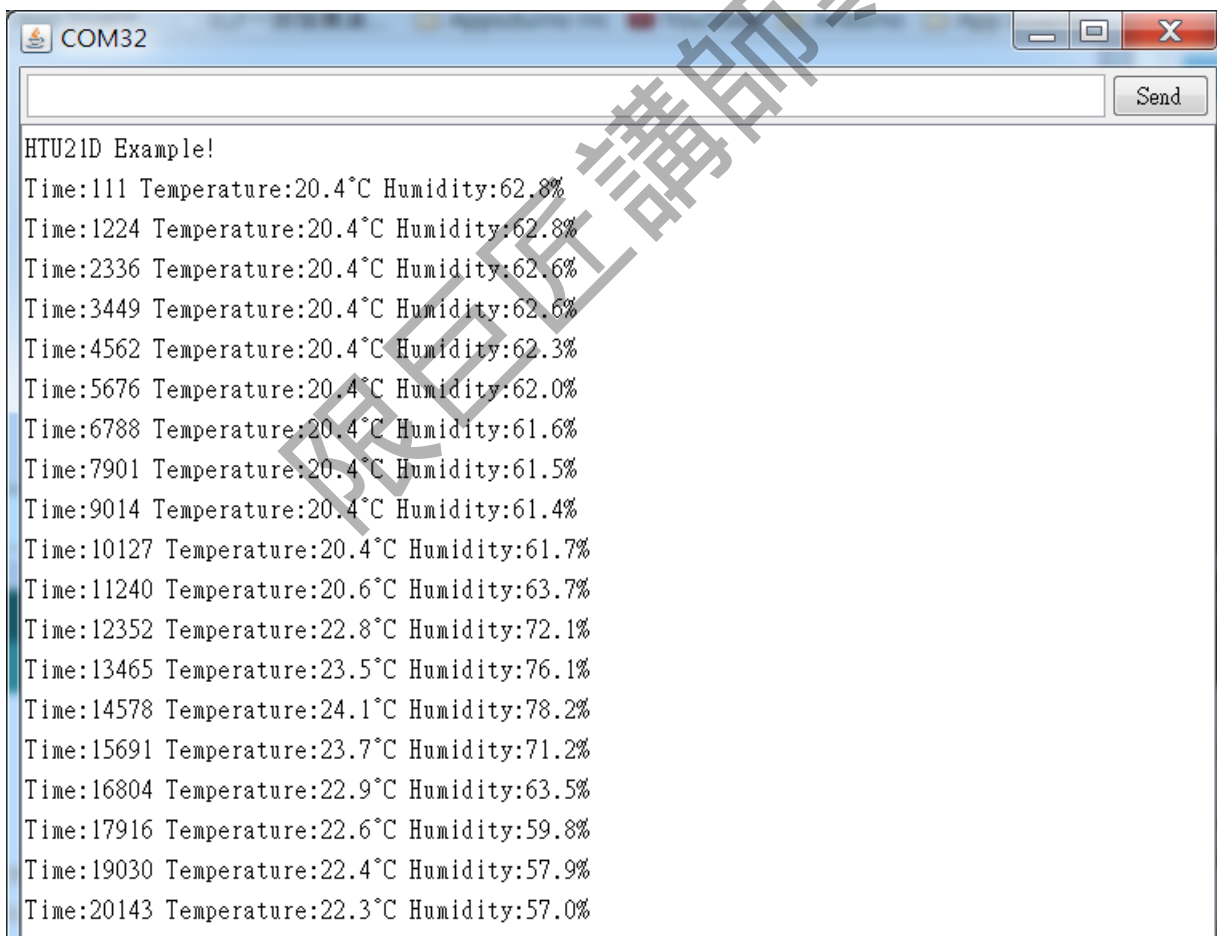
```

Serial.print("Time:");
Serial.print(millis());
Serial.print(" Temperature:");
Serial.print(temp, 1);
Serial.print((char)(176)); // print degree symbol
Serial.print("C");
Serial.print(" Humidity:");
Serial.print(humd, 1);
Serial.print("%");

Serial.println();
delay(1000);
}

```

編譯下載程式後，開啟終端機(9600bps)，立刻看到溫度及濕度數值，使用手摸或熱吹風機吹一下，溫濕度立刻有變化



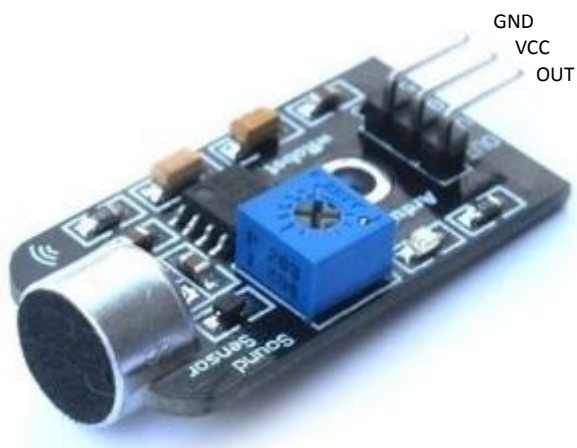
```

COM32
HTU21D Example!
Time:111 Temperature:20.4°C Humidity:62.8%
Time:1224 Temperature:20.4°C Humidity:62.8%
Time:2336 Temperature:20.4°C Humidity:62.6%
Time:3449 Temperature:20.4°C Humidity:62.6%
Time:4562 Temperature:20.4°C Humidity:62.3%
Time:5676 Temperature:20.4°C Humidity:62.0%
Time:6788 Temperature:20.4°C Humidity:61.6%
Time:7901 Temperature:20.4°C Humidity:61.5%
Time:9014 Temperature:20.4°C Humidity:61.4%
Time:10127 Temperature:20.4°C Humidity:61.7%
Time:11240 Temperature:20.6°C Humidity:63.7%
Time:12352 Temperature:22.8°C Humidity:72.1%
Time:13465 Temperature:23.5°C Humidity:76.1%
Time:14578 Temperature:24.1°C Humidity:78.2%
Time:15691 Temperature:23.7°C Humidity:71.2%
Time:16804 Temperature:22.9°C Humidity:63.5%
Time:17916 Temperature:22.6°C Humidity:59.8%
Time:19030 Temperature:22.4°C Humidity:57.9%
Time:20143 Temperature:22.3°C Humidity:57.0%

```



## 聲音感測器介紹與偵測實驗



### ◆ 聲音感測模組簡介

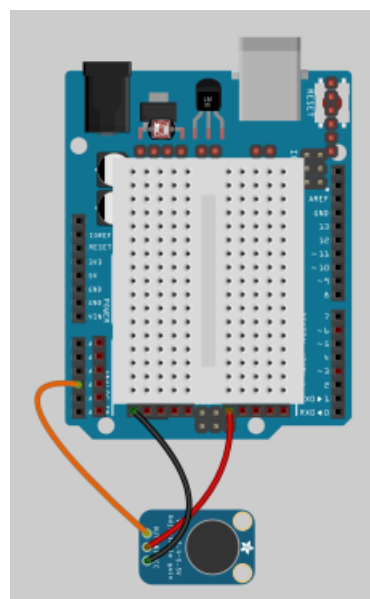
此聲音感測器可偵測聲音大小，其內設計放大電路(LM386)並可調整其增益(Gain)，當數值變化很小時可用起子調整其增益/放大倍數，因此模組輸出類比訊號，所以可接至 Arduino 類比腳位(A0 ~ A5)，利用 analogRead 指令讀取音量大小，並送至 LCD Display 或電腦的串列埠視窗(Serial Monitor) 顯示

### ◆ 特性

- 電壓：+5V
- 類比電壓信號輸出，信號幅度  $VCC/2$
- 內置放大電路，增益可調
- 可透過 A/D 轉換獲得聲音強度的電壓信號

### ◆ 接腳說明

Pin	ID	說明
1	GND	接 GND
2	VCC	接 5V
3	OUT	接類比腳位(A0 ~ A5)，例：A3



## ◆ 程式控制說明

### 測試程式

```
#define Sound_Sensor A3    // Connect Sound sensor to A2 Pin
int sensorValue ; // variable to store the value coming from the sensor

void setup() {
  Serial.begin(9600);
  Serial.println("Starting...");
}

void loop() {
  sensorValue = analogRead(Sound_Sensor);    // 值理論上介於 0~1023 之間
  Serial.println(sensorValue);
  delay(100);
}
```

傳回值理論上介於 0~1023 之間，若逆時針轉到底感覺較為靈敏，傳回值也比較大(max 出現 5~600 的數值)

## 第三堂：RFID 基礎介紹

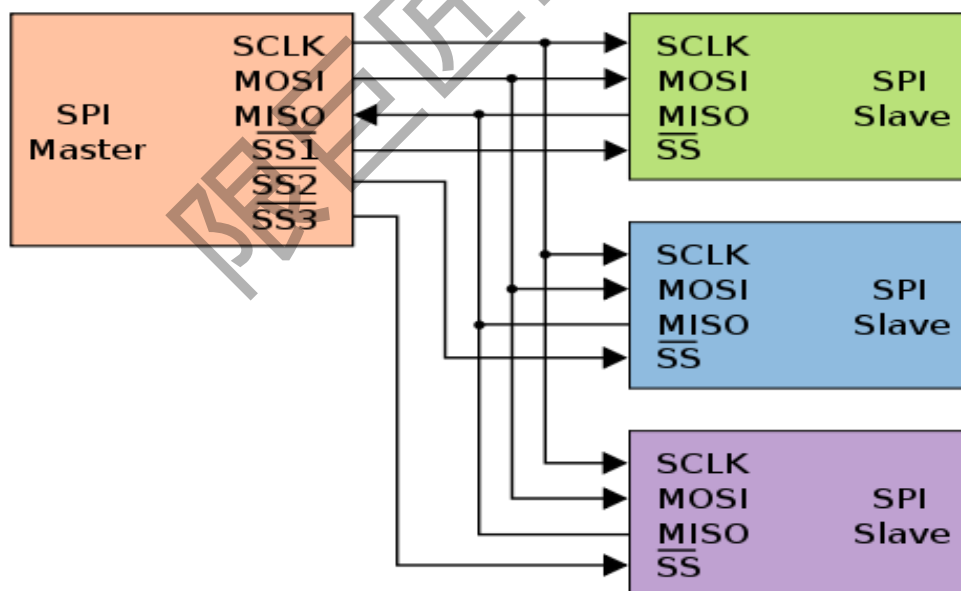
### SPI 傳輸介面簡介

SPI 是 Serial Preipheral Interface 的縮寫，中文名稱是串列週邊介面，該介面是由 Motolor 首先定義，原先是應用在其 68xx 系列的 8 位元處理器上，用以連接 ADC、DAC 轉換、EEPROM、FLASH、RTC、通訊傳輸 IC 以及 DSP/數位信號解碼器之間...等週邊晶片。由於具備有低接腳數，結構單純，傳輸速度快，簡單易用...等特性，目前已經成為業界慣用標準，不只是單晶片微控制器上有，許多新的 SoC 晶片直接就支援多組 SPI 介面，甚至普及到連模組化的產品(如:手機用的 LCD 模組(SDI 介面))。

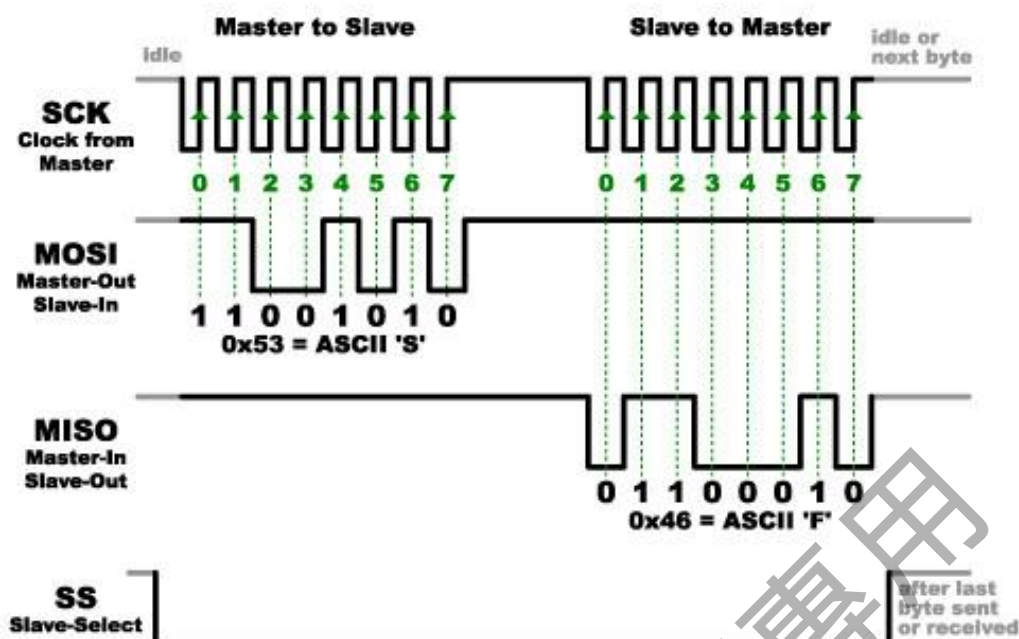
SPI，是一種高速的，全雙工，同步的通信總線，並且只占用 4 個接腳，其通信以主從方式工作，有一個主設備(Master)和一個或多個子設備(Slave)，需要至少 4 條訊號線或 3 條也可以(單向傳輸時)，SPI 匯流排規定了 4 個邏輯訊號介面，如下所述：

- SCLK (Serial Clock)：串行時鐘，由主機發出
- MOSI (Master Output, Slave Input)：主機輸出從機輸入訊號，由主機發出
- MISO (Master Input, Slave Output)：主機輸入從機輸出訊號，由從機發出
- SS (Slave Selected)：片選訊號，由主機發出，低電平有效

其連接示意圖如圖一：

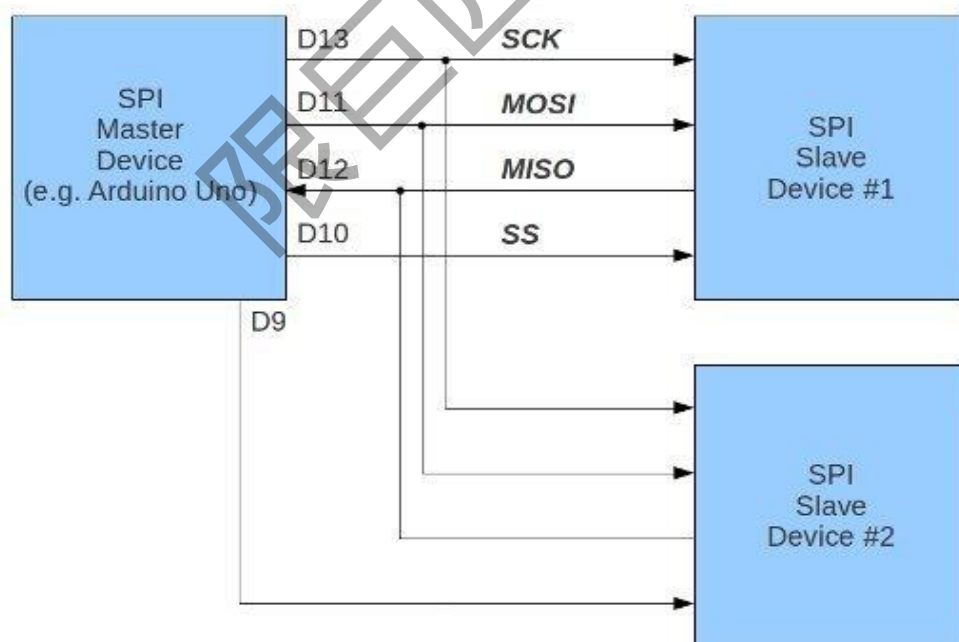


訊號示意圖如圖二：



圖二 SPI 訊號示意圖

而 Arduino 也內建一組 SPI 傳輸介面，其與外界週邊裝置的连接則如圖三所示：



圖三 Arduino UNO 與 SPI 周邊連接示意圖



## RC522-RFID 控制板介紹

RFID-RC522 控制板採用 NXP MF RC522 晶片設計讀卡電路，使用方便，適用於設備開發、讀卡器開發等應用的使用者及需要進行射頻卡終端設計/生產的使用者。模組採用電壓為 3.3V, 通過 SPI 傳輸介面，就可以直接與任何具備 SPI 介面的 CPU 主機板相連接與通信



### ◆ RC522 晶片簡介

MF RC522 是應用於 13.56MHz 非接觸式通信中高集成度的讀寫卡晶片，是 NXP 公司推出的一款低電壓、體積小的非接觸式讀寫卡晶片，是智慧型儀表和可攜式手持設備研發的理想選擇。MF RC522 利用了先進的調變和解調概念，集成了在 13.56MHz 下所有類型的被動非接觸式通信方式和協定。支援 14443A 相容應答器信號。數位部分處理 ISO14443A Frame(幀)和錯誤檢測。此外，還支援快速 CRYPTO1 加密演算法。MFRC522 支援 MIFARE 系列更高速的非接觸式通信，雙向資料傳輸速率高達 424kbit/s。作為 13.56MHz 高集成度讀寫卡系列晶片家族的成員，MF RC522 與 MF RC500 和 MF RC530 有不少相似之處，同時也具備許多特點和差異。它與主機間通信採用 SPI 模式，有利於減少連線，縮小 PCB 板體積，降低成本。

### 電氣參數簡介

工作電流：13—26mA/直流 3.3V

空閒電流：10-13mA/直流 3.3V

休眠電流：<80uA

峰值電流：<30mA

工作頻率：13.56MHz

支援的卡類型：mifare1 S50、mifare1 S70、mifare UltraLight、mifare Pro、mifare Desfire

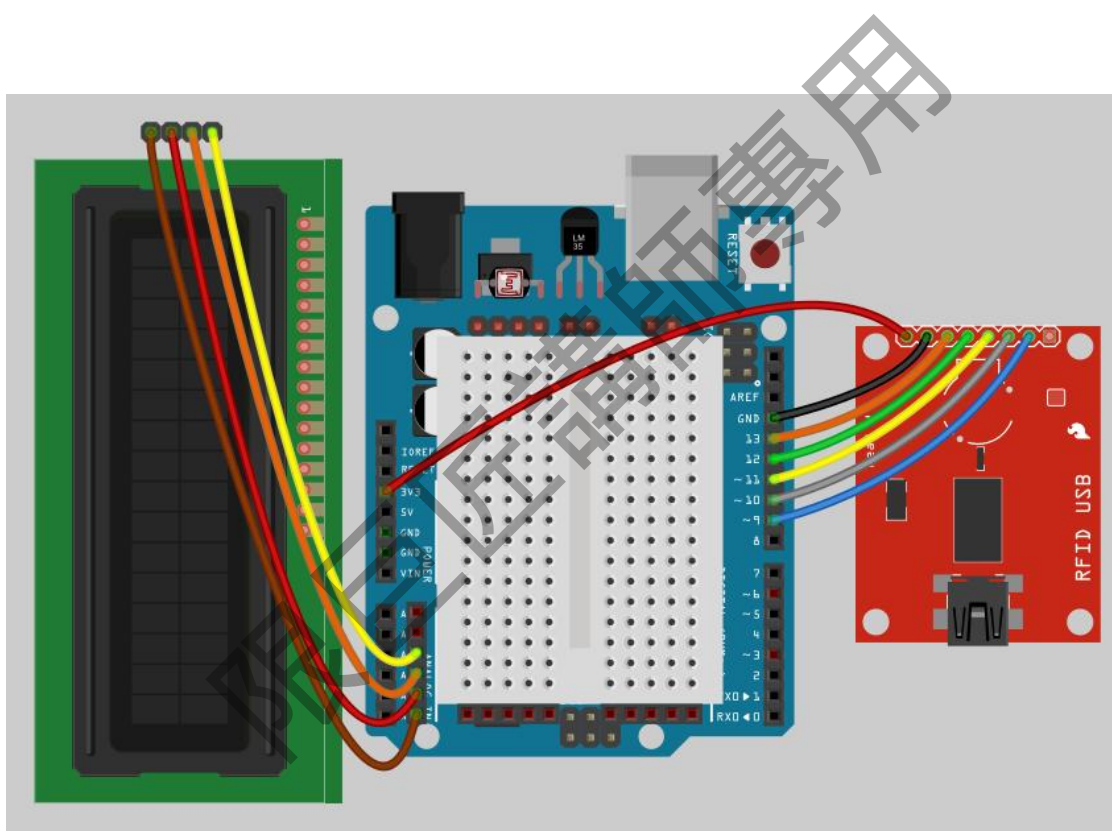
產品物理特性：尺寸：40mm×60mm

### 模組介面 SPI 參數

資料傳輸速率：最大 10Mbit/s

◆ 接腳說明(採用 SPI 介面)

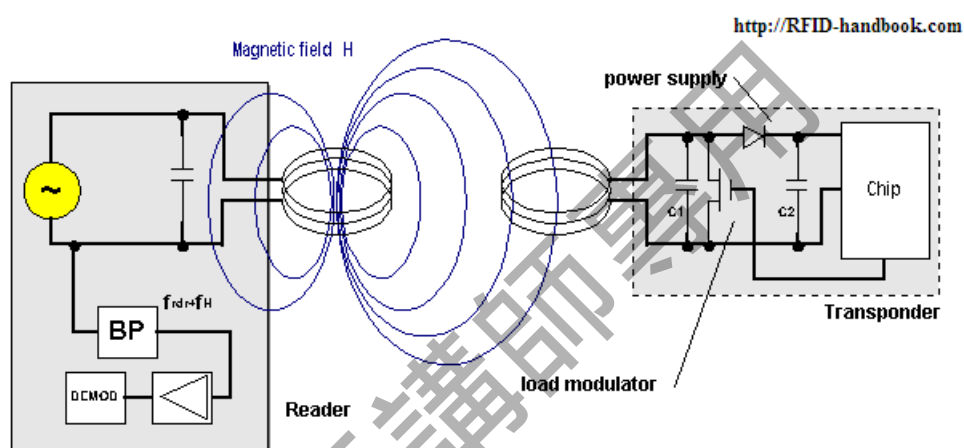
接腳	名稱	RFID Reader(SPI 介面)
1	IRQ	X
2	RST	接Arduino D9
3	NSS	接 Arduino D10
4	MOSI	接 Arduino D11
5	MISO	接 Arduino D12
6	SCK	接 Arduino D13
7	GND	接GND
8	3.3V	接 <b>3.3V</b>



RFID 對應接腳接線圖

## RFID 的運作原理

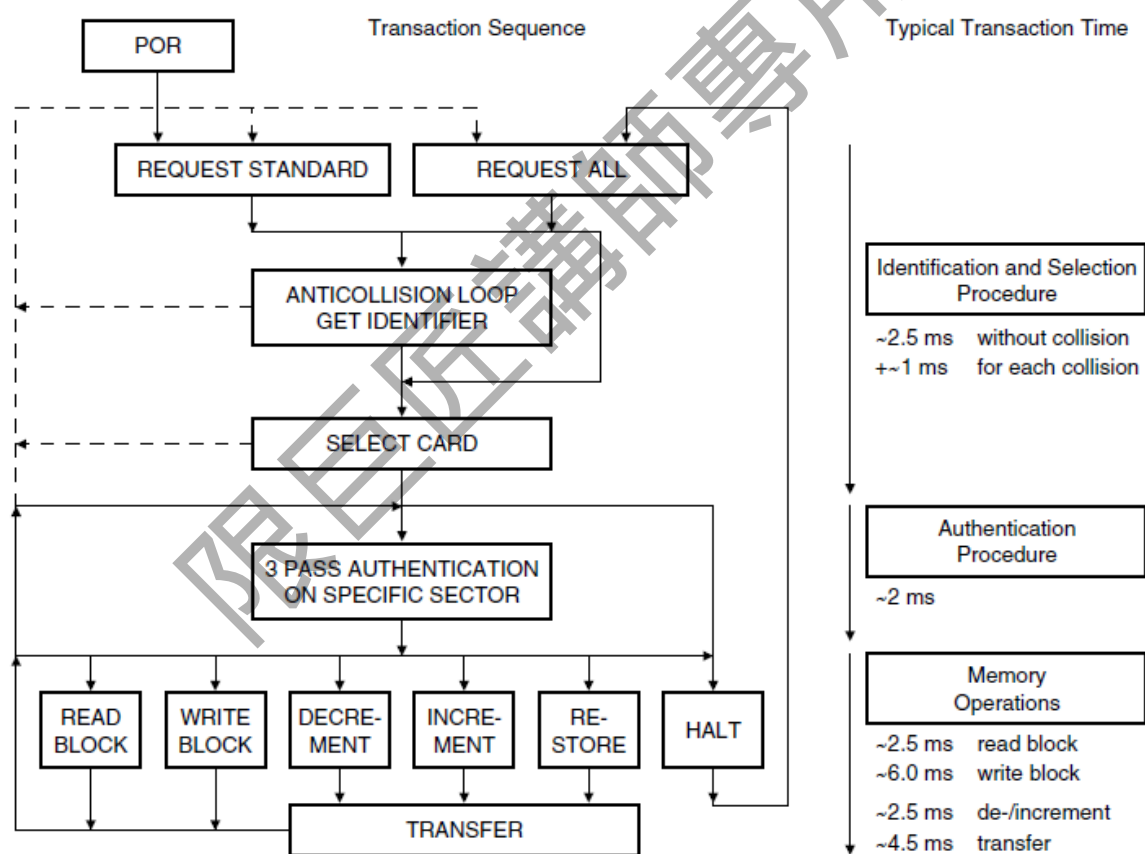
RFID(Radio Frequency Identification)是非接觸的自動識別技術，利用射頻訊號與空間電磁感應進行通訊，達到自動識別被標示物體的目的。系統包含一個讀寫器(Reader)與 Tag，讀寫器向 RFID Tag(M1 卡)發一組固定頻率的電磁波，卡片內有一個 LC 串聯諧振電路，其頻率與讀寫器發射的頻率相同，在電磁波的激勵下，LC 諧振電路產生共振，從而使電容內有了電荷，在這個電容的另一端，接有一個單向導通的二極體(Diode)，將電容內的電荷 送到另一個電容內儲存，當所積累的電荷達到一定電壓 ( $\sim 2V$ )時，此電容可做為 MF1S50 IC 的控制單元的電源，並根據相應區的有效存取位元(access bits)來控制將卡內資料發射出去或讀取讀寫器的資料。其運作程序如下所述：



- 請求應答(Answer to request)  
RFID Tag 的通訊協定和通訊串列傳輸速率是定義好的，當有卡片進入讀寫器的操作範圍時，讀寫器會以特定的協定與它通訊，從而確定該卡是否為相容的卡，即驗證卡片的類型。而卡片也會依請求代碼發送回應 ATQA 訊號
- 防衝突機制 (Anticollision Loop)  
當有多張卡進入讀寫器操作範圍時，防衝突機制會從其中選擇一張進行操作，未選中的則處於準備模式等待下一次選卡，該過程會返回被選卡的序號。
- 選擇卡片(Select Tag)  
讀寫器使用選擇卡命令，選擇其中一張卡的序號進行確認和記憶體相關操作，此時卡回傳 ATS 碼(Answer To Select =08h)，讀寫器透過 ATS 可以確定被選中的卡的類型，並同時返回卡的容量代碼。
- 三次互相授權認證機制(3 Pass Authentication)  
選定要處理的卡片之後，讀寫器就確定要存取的區塊號碼(Block #)，並對該區塊密碼進行密碼校驗，在三次相互認證之後就可以通過加密進行通訊。

- 區塊(Block)的操作方式(在執行任何區塊操作前，卡必須先要被選擇並經過碼驗證機制)

方式	操作	描述	區塊類型(Block type)
讀 (Read)	讀一個區塊(Block)	資料、數值與 Trailer 區塊	資料、數值與 Trailer 區塊
寫 (Write)	寫一個區塊(Block)	資料、數值與 Trailer 區塊	資料、數值與 Trailer 區塊
加 (Increment)	對數值區塊(value block)進行加值	數值區塊(value block)	數值區塊(value block)
減 (Decrement)	對數值區塊進行減值	數值區塊(value block)	數值區塊(value block)
存儲 (Restore)	將區塊中的內容存到資料暫存器中	數值區塊(value block)	數值區塊(value block)
傳輸 (Transfer)	將資料暫存器中的內容寫入區塊中	數值區塊(value block)	數值區塊(value block)

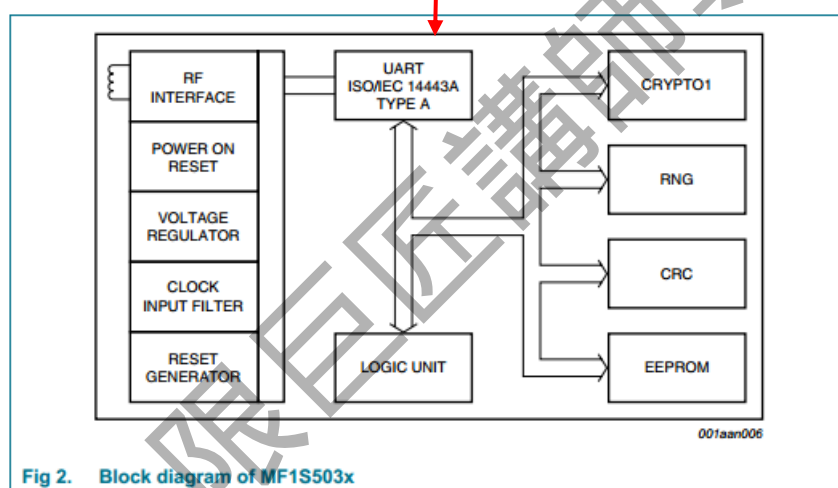
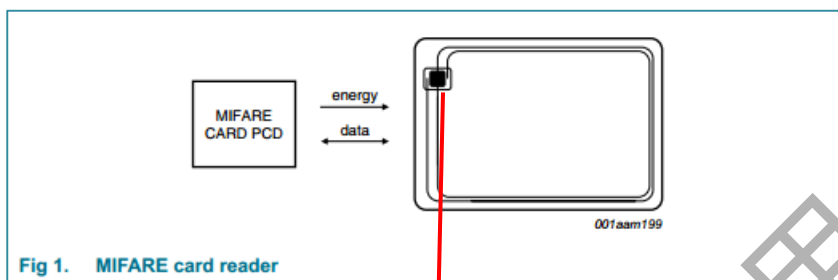


RFID Tag 的運作程序圖

## RFID Tag 介紹

目前市面上有許多不同形式的 RFID 標籤(Tag)，每個 Tag 上有唯一的序號或 ID(4 Bytes)可供辨識，其內部也含有不同容量的記憶體可供讀寫(1K/4K Bytes,...)。

圖一為 Mifare Reader 與 Tag 通訊示意圖，圖二則為標籤(MF1S503X)內控制晶片的方塊圖。

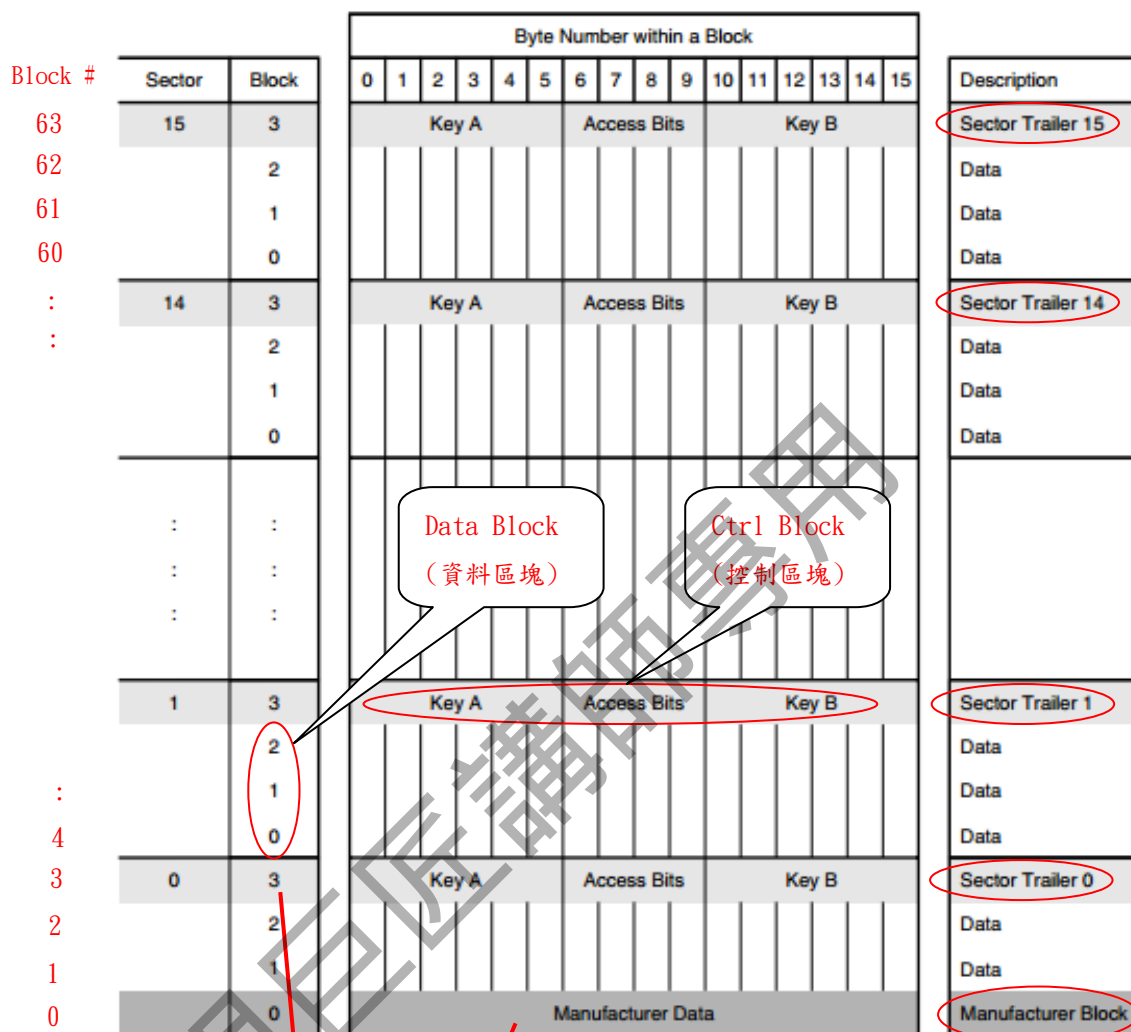




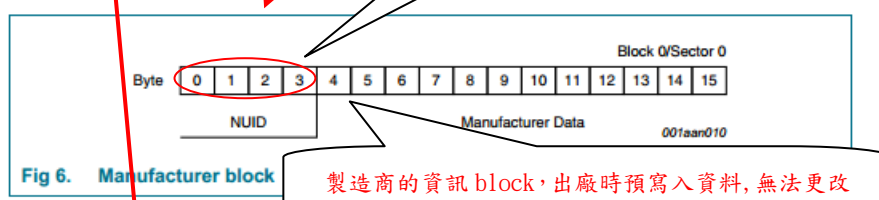
◆ Mifare Classic Tag(MF1S50)/1K bytes 記憶體資料格式(16 Sec x 4 Blk x 16 bytes)

### Memory organization

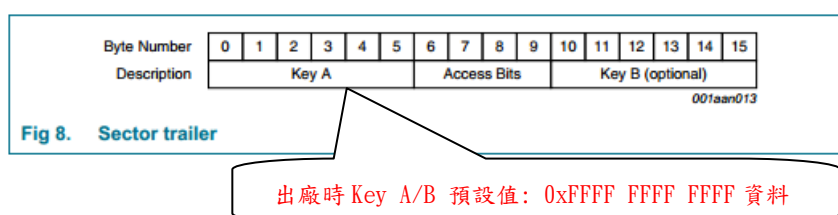
The 1024 × 8 bit EEPROM memory is organized in 16 sectors of 4 blocks. One block contains 16 bytes.



Manufacturer block



Sector trailer (block 3)



## ◆ 記憶體區塊簡介

– 記憶體區塊(Block) 有三種型態：

1. 讀、寫區塊：一般的資料保存，可以進行讀、寫操作
2. 數值區塊(value block)：用作資料值(4 Bytes)，可以進行加值、減值、讀值操作，例如電子錢包應用

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	value				value				value				adr	$\overline{\text{adr}}$	adr	$\overline{\text{adr}}$

3. 控制區塊：每個扇區的區塊 3 為控制區塊，包括密碼 A (6 位元組)、存取控制 (4 位元組)、密碼 B (6 位元組)，如下圖：

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Key A						Access Bits				Key B (optional)					

- 每個扇區(Sector)的密碼和存取控制都是獨立的，可以根據實際需要設定各自的密碼及存取控制。存取控制為 4 個位元組，共 32 位元，扇區中的每個區塊 (包括資料區塊和控制區塊) 的存取條件是由密碼和存取控制共同決定的，在存取控制中每個區塊都有對應的三個控制位元，其結構與對應之操作權限如下圖示：

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Key A						Access Bits				Key B (optional)					

	Bit 7	6	5	4	3	2	1	0
Byte 6	$\overline{\text{C2}}_3$	$\overline{\text{C2}}_2$	$\overline{\text{C2}}_1$	$\overline{\text{C2}}_0$	$\overline{\text{C1}}_3$	$\overline{\text{C1}}_2$	$\overline{\text{C1}}_1$	$\overline{\text{C1}}_0$
Byte 7	$\text{C1}_3$	$\text{C1}_2$	$\text{C1}_1$	$\text{C1}_0$	$\overline{\text{C3}}_3$	$\overline{\text{C3}}_2$	$\overline{\text{C3}}_1$	$\overline{\text{C3}}_0$
Byte 8	$\text{C3}_3$	$\text{C3}_2$	$\text{C3}_1$	$\text{C3}_0$	$\text{C2}_3$	$\text{C2}_2$	$\text{C2}_1$	$\text{C2}_0$
Byte 9	user data							

Access Bits	Valid Commands		Block	Description
$\text{C1}_3 \text{ C2}_3 \text{ C3}_3$	read, write	→	3	sector trailer
$\text{C1}_2 \text{ C2}_2 \text{ C3}_2$	read, write, increment, decrement, transfer, restore	→	2	data block
$\text{C1}_1 \text{ C2}_1 \text{ C3}_1$	read, write, increment, decrement, transfer, restore	→	1	data block
$\text{C1}_0 \text{ C2}_0 \text{ C3}_0$	read, write, increment, decrement, transfer, restore	→	0	data block

而各區塊對應的存取位元定義如下：

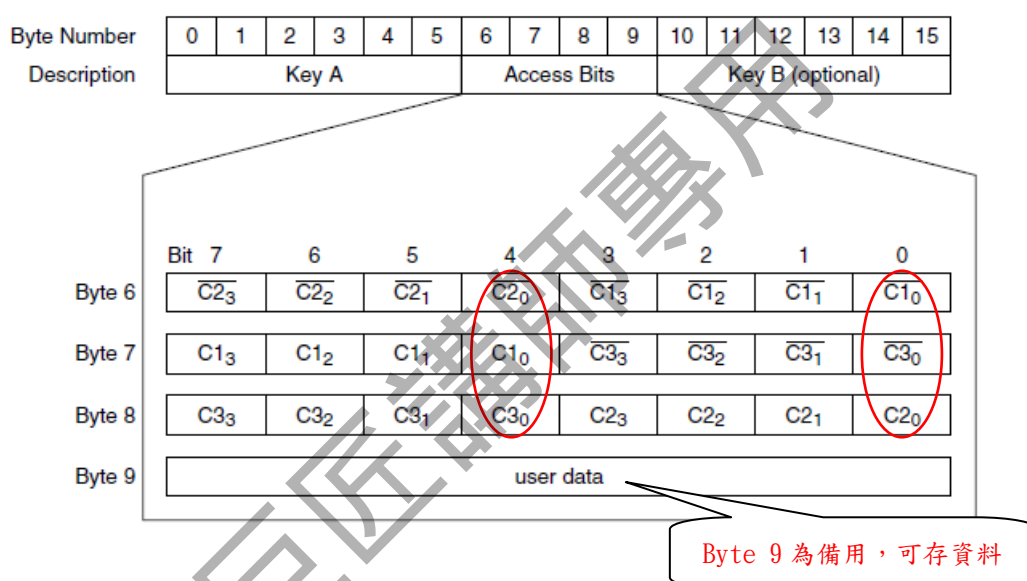
區塊 0：C10 C20 C30

區塊 1：C11 C21 C31

區塊 2：C12 C22 C32

區塊 3：C13 C23 C33

三個控制位元以正和反兩種形式存在於存取控制位元組中，決定了該區塊的存取權限（如進行減值操作必須驗證 KEY A，進行加值操作必須驗證 KEY B，等等）。三個控制位元在存取控制位元組中的位置，以區塊 0 為例：



– 資料區塊（0、1、2）的存取控制如下：

Access bits			Access condition				Application
C1	C2	C3	Read	Write	Increment	Dec/Trans/Rest	
0	0	0	key A B	key A B	key A B	key A B	transport
0	1	0	key A B	never	never	never	read/write block
1	0	0	key A B	key B	never	never	read/write block
1	1	0	key A B	key B	key B	key A B	value block
0	0	1	key A B	never	never	key A B	value block
0	1	1	key B	key B	never	never	read/write block
1	0	1	key B	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

（KeyA|B 表示密碼 A 或密碼 B，Never 表示任何條件下不能實現）

從表中可以看出：

C1C2C3=000(出廠預設值)時最寬鬆，驗證密碼 A 或密碼 B 後可以進行任何操作

C1C2C3=111 無論驗證哪個密碼都不能進行任何操作，相當於把對應的區塊凍結了

C1C2C3=010 和 C1C2C3=101 都是唯讀，如果對應的資料區塊寫入的是只能讀取(Read only)的訊息，可設為這兩種模式

C1C2C3=001 時只能讀和減值，電子錢包一般設為這種模式，比如公車與捷運的悠遊卡或一卡通，使用者只能查詢或扣錢，不能私下加值，加值的時候透過儲值設備，先改變控制位元使卡片可以充值，充完值會再改回來。

- 控制區塊 3 的存取控制與資料區塊(0、1、2) 不同，它的存取控制如下：

Access bits			Access condition for						Remarks
			KEY A		Access bits		Key B		
C1	C2	C3	Read	Write	Read	Write	Read	Write	
0	0	0	never	key A	key A	never	key A	key A	Key A is able to read key B
0	1	0	never	never	key A	never	key A	never	Key A is able to read key B
1	0	0	never	key B	key A B	never	never	key B	
1	1	0	never	never	key A B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key A is able to read key B
0	1	1	never	key B	key A B	key B	never	key B	
1	0	1	never	never	key A B	key B	never	never	
1	1	1	never	never	key A B	never	never	never	

從上表中可以看出：

密碼 A 是永遠也讀不出來的，如果用戶的資料區塊指定了驗證密碼 A 卻忘了密碼 A，也就意味著這個資料區塊作廢了，但本扇區其他資料區塊和其他扇區的資料區塊不受影響

存取控制的寫控制在設置時一定要小心，一旦弄成了“Never”，則整個扇區的存取條件再也無法改變

**C1C2C3=001(出廠預設值)時最寬鬆**，除了密碼 A 不能讀之外，驗證了密碼 A 其他讀寫操作都可以進行

當 C1C2C3=000、C1C2C3=010 和 C1C2C3=001 時，所有的操作都不使用密碼 B，這時候密碼 B 佔據的 6 個位元組可以提供給使用者作為普通資料存儲用

由於**出廠時資料區塊控制位元的預設值是 C1C2C3=000**，控制區塊的預設值是 C1C2C3=001，而 Byte9 一般是 69H，所以出廠白卡的控制位元組通常是 **FF078069H**。

## RFID Library API 使用說明

- **RFID 類別的全域變數(global variable)**  
unsigned char **serNum[5]** : 紀錄 RFID Tag 的 ID/序號(4 Bytes + 1 byte(檢查碼))  
**serNum [0..3]**: serial number of the card  
**serNum [4]**: XOR checksum of the serNum [0..3]
- **RFID(int chipSelectPin, int NRSTPD)** : RFID class constructor  
ex : RFID rfid(10, 9); //宣告一個 RFID 類別的物件，名稱為 rfid 且其 NSS(CS) 與 RST Pin 分別接至 Arduino 的 D10 與 D9 腳位
- **bool isCard()** : 檢查是否偵測到 RFID Tag ? 並回傳 true/false  
ex : if (rfid.isCard()) {  
      ..... } //若偵測到 RFID Tag 則處理下一步
- **void init()** : 初始化 RFID  
ex : rfid.init() ; //將 RFID 物件 rfid 初始化
- **bool readCardSerial()** : 讀取 RFID 的 ID/序號(4 bytes)與檢查碼(1 byte)  
ex : if (rfid.readCardSerial()) {.... } ; //可用 rfid.serNum[0].. rfid.serNum[4] 取出序號
- **unsigned char anticoll(unsigned char \*serNum)** : 處理當有多張卡進入讀寫器操作範圍時，防衝突機制會從其中選擇一張進行操作，並回傳處理狀態(status)與將此張卡的序號儲存於 serNum[0]~ serNum[3] Array 內  
回傳處理狀態(status)有 :
  - MI\_OK : 成功
  - MI\_NOTAGERR
  - MI\_ERRex : status = rfid.anticoll(rfid.serNum);  
if (status == MI\_OK) {.... } ; //可用 rfid.serNum[0].. rfid.serNum[4] 取出序號
- **unsigned char SelectTag(unsigned char \*serNum)** : 選擇序號為 serNum[0..3]的卡片並回傳該卡片的記憶體容量(單位 : Bits)，當要處理卡片記憶體區塊的任何動作(讀/寫/加/減,..)前，皆須先透過此程序選擇一張卡片  
ex: RC\_size = rfid.SelectTag(rfid.serNum); // 選擇一張卡並回傳卡片的容量  
if (RC\_size != 0) {.... } //可印出卡片的容量(單位 : Bits)



- `unsigned char auth(unsigned char authMode, unsigned char BlockAddr, unsigned char *Sectorkey, unsigned char *serNum)` : 處理卡片記憶體區塊的任何動作(讀/寫/加/減,...)前, 皆須先透過密碼的驗證程序, 通過後才能操作記憶體區塊資料  
輸入參數如下 :

`authMode` : 驗證密碼模式, PICC\_AUTHENT1A 指使用密碼 A(Key A)驗證, PICC\_AUTHENT1B 指使用密碼 B(Key B)

`BlockAddr` : 區塊編號(0 ~ 63)

`Sectorkey` : 密碼儲存位址

`serNum` : 卡片序號儲存位址

回傳處理狀態(status) :

- `MI_OK` : 成功
- `MI_NOTAGERR`
- `MI_ERR`

```
ex : status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],  
                        rfid.serNum); // Authentication by verify Key A  
    if (status == MI_OK) { Read or write block ,... }
```

- `unsigned char read(unsigned char blockAddr, unsigned char *recvData)` : 讀取 RFID 所指定 blockAddr(0 ~ 63)的整個區塊資料(16 bytes)並將其儲存於 `receData` 的字元串中

ex : `rfid.read(11, receData)` ; //讀取 RFID Tag block 11 的資料(16 bytes)

- `unsigned char write(unsigned char blockAddr, unsigned char *writeData)` : 將 `writeData` 字元字串(16 Bytes)中的資料寫入 RFID 所指定的 blockAddr

ex : `rfid.read(11, writeData)` ; //將 writeData 的資料(16 bytes)寫入 Tag 的 block 11

- `void halt()` : 讓卡片進入休眠狀態  
ex : `rfid.halt()` ; //設定卡片進入休眠狀態

## Tag ID 讀取實驗(cardRead. ino)

```
#include <SPI.h>
#include <RFID.h>

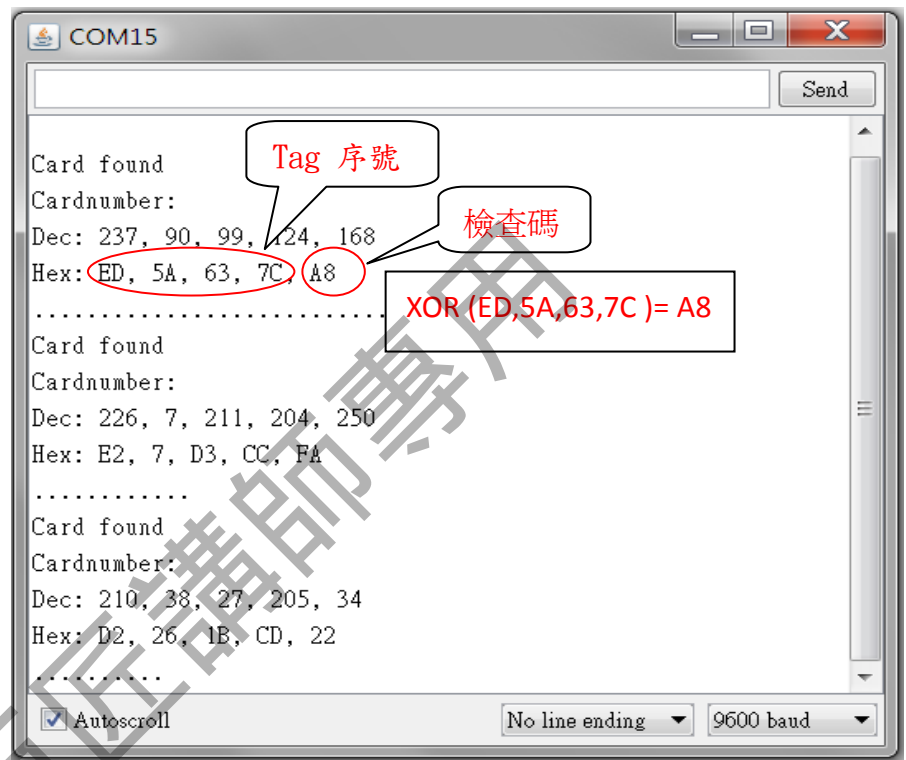
#define SS_PIN 10
#define RST_PIN 9

RFID rfid(SS_PIN, RST_PIN);

// Setup variables:
int serNum0;
int serNum1;
int serNum2;
int serNum3;
int serNum4;

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
}

void loop()
{
  if (rfid.isCard()) {
    if (rfid.readCardSerial()) {
      if (rfid.serNum[0] != serNum0
          && rfid.serNum[1] != serNum1
          && rfid.serNum[2] != serNum2
          && rfid.serNum[3] != serNum3
          && rfid.serNum[4] != serNum4
      ) {
        /* With a new cardnumber, show it. */
        Serial.println(" ");
        Serial.println("Card found");
        serNum0 = rfid.serNum[0];
```



```

serNum1 = rfid.serNum[1];
serNum2 = rfid.serNum[2];
serNum3 = rfid.serNum[3];
serNum4 = rfid.serNum[4];

//Serial.println(" ");
Serial.println("Cardnumber:");
Serial.print("Dec: ");
Serial.print(rfid.serNum[0], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[1], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[2], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[3], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[4], DEC);
Serial.println(" ");

Serial.print("Hex: ");
Serial.print(rfid.serNum[0], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[1], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[2], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[3], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[4], HEX);
Serial.println(" ");
} else {
    /* If we have the same ID, just write a dot. */
    Serial.print(".");
}
}

rfid.halt();
delay(200) ; // delay 200 ms
}

```

## ◆ RFID 讀寫 Block(模擬儲值、消費、充值的情境)實驗

讀寫主要流程如下：

1. 尋找卡片，回傳**卡片型態**代號
2. 呼叫防衝撞程序(Anti-collision loop in readCardSerial())，**回傳卡片序號**
3. 選擇作用卡片
4. **寫入新密碼**到區塊 11(控制區塊)，**寫入數值 100** 到區塊 8(資料區塊)的位址
5. **讀取**區塊 8 的資料
6. 讀取區塊 8，減 18，運算完成將結果重新寫入
7. 讀取區塊 8，加 10，運算完成將結果重新寫入
8. **中止**卡片存取，進入睡眠模式

```

/*
 * 文件 名：RFID_blockRW_demo.ino
 * 功能描述：Mifare1 尋卡→防衝突→選卡→讀寫 介面
 */

/*
 * Documentation：RFID_blockRW_demo.ino
 * Created By：Evan WWW.B2CQSHOP.COM
 * Creation Date：2011.09.19
 * Modified：
 * Translated By：Gareth Halfacree gareth.halfacree.co.uk
 * Description：Mifare, Find Cards → Anti-Collision → Read and Write Cards
 * Translated By：Tony Su/2015/04/16 www.appsduino.com
 * Description：simplified with rfid library
 */
// the sensor communicates using SPI, so include the library:
#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

RFID rfid(SS_PIN, RST_PIN);

// Maximum length of an array
#define MAX_LEN 16

```

```

unsigned char  writeData[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100}; // Initialise
$100
unsigned char  moneyConsume = 18 ; // Spend $18
unsigned char  moneyAdd = 10 ; // Recharge $10

// Sector A Password, 16 sectors, each sector password is 6-bytes
unsigned char sectorKeyA[16][16] = //舊密碼
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, //當改變密碼後,需使用新密碼
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
};
unsigned char sectorNewKeyA[16][16] = //新密碼
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x84, 0x07, 0x15, 0x76, 0x14},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x33, 0x07, 0x15, 0x34, 0x14},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x33, 0x07, 0x15, 0x34, 0x14},
};

void setup() {
  Serial.begin(9600);
  // start the SPI library:
  SPI.begin();
  rfid.init();
  delay(1000) ;
  Serial.println("RFID write/read block Example");
  Serial.println("Demo for Read/Write data in (Sector 2,Block 0) or block #8");
}

void loop()
{
  unsigned char i,tmp;
  unsigned char status;
  unsigned char str[MAX_LEN];
  unsigned char RC_size;
  unsigned char blockAddr; // Select the operating block address, 0-63

```



```
//-----
if (rfid.isCard()) { // 檢查是否偵測到 RFID Tag ? 並回傳 true/false
  if (rfid.readCardSerial()) { //讀取 RFID 的 ID/序號(4 bytes)與檢查碼(1 byte)
    /* With a new cardnumber, show it. */
    Serial.println(" ");
    Serial.println("Card found");
    Serial.print("Serial #(Hex) : ");
    Serial.print(rfid.serNum[0], HEX);
    Serial.print(", ");
    Serial.print(rfid.serNum[1], HEX);
    Serial.print(", ");
    Serial.print(rfid.serNum[2], HEX);
    Serial.print(", ");
    Serial.print(rfid.serNum[3], HEX);
    Serial.print(", ");
    Serial.print(rfid.serNum[4], HEX);
    Serial.println(" ");
  }
}

//-----

// Election card, return capacity
RC_size = rfid.SelectTag(rfid.serNum); // 選擇一張卡並回傳卡片的容量
// Now a card is selected. The UID and SAK is in mfrc522.uid.
if (RC_size != 0) {
  Serial.print("The size of the card is : ");
  Serial.print(RC_size, DEC);
  Serial.println("K bits");//單位 : K bits
}

// 存入$100 in Registration card
blockAddr = 11; //驗證 Data block 11 中的密碼 A
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],
  rfid.serNum);// Authentication by verify Key A
if (status == MI_OK) {
  // Write data
  status = rfid.write(blockAddr, sectorNewKeyA[blockAddr/4]);//寫入新的密碼
  Serial.print("set the new card password, and can modify the data of the Sector ");
  Serial.print(blockAddr/4, DEC);
  Serial.println(" : ");
}
```

```
for (i=0; i<6; i++) {
    Serial.print(sectorNewKeyA[blockAddr/4][i], HEX);
    Serial.print(" , ");
}
Serial.println(" ");
blockAddr = blockAddr - 3 ;
status = rfid.write(blockAddr, writeData); //在 block #8 的 byte #15 存入$100
if(status == MI_OK)
    Serial.println("You are CyberShop VIP Member, The card has $100 !");
}

// Card reader
blockAddr = 11;          //驗證 Data block 11 中的密碼 A
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
    rfid.serNum);        // Authentication
if (status == MI_OK) {
    // Read data
    blockAddr = blockAddr - 3 ;
    status = rfid.read(blockAddr, str); //讀取 Sector 2 的控制區塊資料(block #11)
    if (status == MI_OK) {
        Serial.println("Read from the card , the data is : ");
        for (i=0; i<16; i++)
        {
            Serial.print(str[i], DEC);
            Serial.print(" , ");
        }
        Serial.println(" ");
    }
}
}
```

```
// 模擬消費/Consumer
blockAddr = 11;          //驗證 Data block 11 中的密碼 A
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
    rfid.serNum);    // Authentication
if (status == MI_OK) {
    // Read data
    blockAddr = blockAddr - 3 ;
    status = rfid.read(blockAddr, str);
    if (status == MI_OK) {
        if( str[15] < moneyConsume )
            Serial.println(" The money is not enough !");
        else {
            str[15] = str[15] - moneyConsume;
            status = rfid.write(blockAddr, str); //在 block #8 的 byte #15 減去$18 後再存入
            if(status == MI_OK) {
                Serial.print("You pay $18 for items Now, Your money balance is :  $");
                Serial.print(str[15], DEC);
                Serial.println(" ");
            }
        }
    }
}

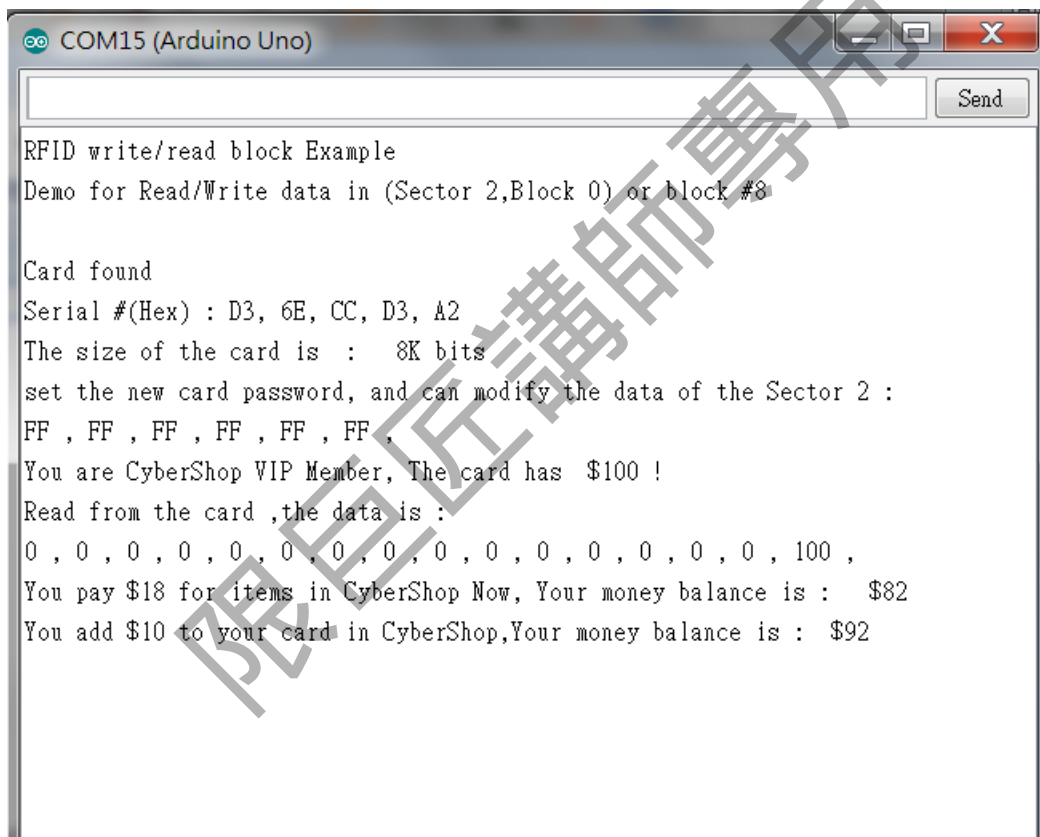
// Recharge
blockAddr = 11;          // Data block 11
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
    rfid.serNum);    // Authentication
if (status == MI_OK)
{
    // Read data
    blockAddr = blockAddr - 3 ;
    status = rfid.read(blockAddr, str);
    if (status == MI_OK) {
        tmp = (int)(str[15] + moneyAdd) ;
        //Serial.println(tmp, DEC);
        if( tmp > 254 ) //一個 Byte 數值不能超過 255
            Serial.println(" The money of the card can not be more than 255 !");
        else {
            str[15] = str[15] + moneyAdd ;
            status = rfid.write(blockAddr, str);
        }
    }
}
```

```

        if(status == MI_OK){
            Serial.print("You add $10 to your card,Your money balance is : $");
            Serial.print(str[15],DEC);
            Serial.println(" ");
        }
    }
}

rfid.halt(); // Enter Sleep Mode
delay(500) ; // waiting for 0.5 sec
}

```



範例程式執行結果

## 第 4 堂：物聯網無線通訊篇(介紹與實驗)

### 藍牙主從模組介紹(HC-05)

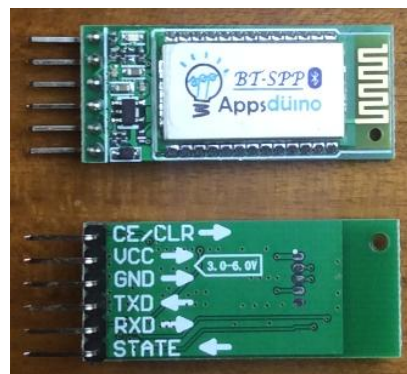
此藍牙模組使用 UART(Tx/Rx)介面來收發訊息，並能與藍牙手機或平板透過無線通訊

預設值：

Baud rate：57,600bps

ID：Appsdüino

PIN Code(配對密碼)：0000



其腳位定義如下：

Pin	ID	說明
1	CE/CLR	X(沒接)
2	VCC	3 ~ 6V
3	GND	
4	TXD	UART TXD(接至 UNO RX Pin/D0)
5	RXD	UART RXD(接至 UNO TX Pin/D1)
6	STATE	X(重置原先之設定值, 正常沒接)

備註：因為藍牙與 USB-Serial 共用相同序列埠 (D0, D1 Pin)，因此上傳程式時，請先拔除藍牙





## 修改藍牙模組名稱

傳送：AT+NAME=**name**

回傳：OK

**name**：所要設置的名稱，需 20 個字元以內。

例：傳送AT+NAME=**Dino928**

回傳：OK

這時藍牙名稱改為Dino928

## 查詢藍牙模組名稱

傳送：AT+NAME?

回傳：+NAME:**name**

**OK**

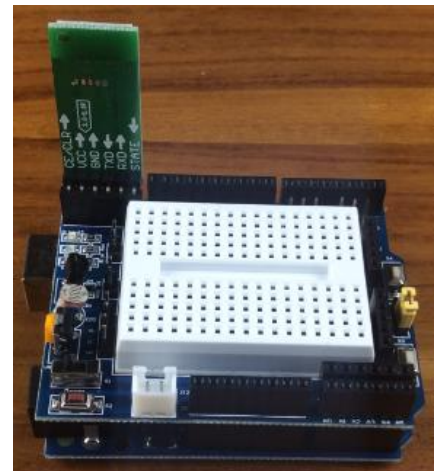
例：傳送AT+NAME=Appsduino

回傳：OK

傳送：AT+NAME?

回傳：+NAME: Appsduino

OK



限巨匠講師專用

### 修改藍牙模組主從角色(Master/Slave)

傳送：AT+ROLE=<Role>

//Role = 0 : Slave(被動連接)

回傳：OK

//Role = 1 : Master(主動搜尋周圍 SPP 設備並

連接)

### 查詢藍牙模組主從角色(Master/Slave)

例：傳送AT+ROLE?

回傳：+ROLE:<Role>

//Role = 0 : Slave

OK

//Role = 1 : Master

### 藍牙模組 Baud rate 修改及速率對應表

指令：AT+UART=<BaudRate>,<StopBits>,<ParityBits>

回傳：OK

例：要設定 Bluetooth 成 9600 bps，則需送 "AT+UART=9600" command to Bluetooth module.

若成功則回送"OK" 訊息！

Note 1：設定時需注意雙方的 Baud Rate 需一致

Note 2：此 BT 模組雖支援至 1,382,400bps，但 Arduino UNO 控制板之 UART 僅支援至 115,200bps

Note 3：預設之資料格式為 8 data bits, no parity, one stop bit

限巨匠講師專用

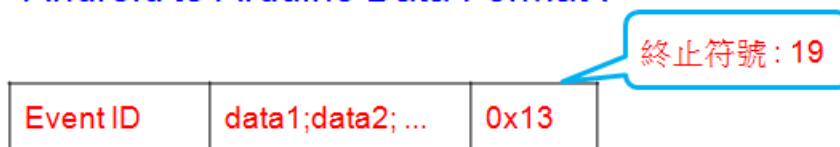
設定/查詢傳輸速率(Baud Rate)AT 指令	回應(Response)	備註
AT+UART=<BaudRate>, <StopBits>, <ParityBits>	OK	BaudRate：串列傳輸速率 (bits/s) 如下：
AT+UART?	+ UART=<BaudRate>, <Stop>, <Parity>	1200 2400 4800 9600 19200 38400 57600 115200 Stop Bits：停止位 0：1 位 1：2 位 Parity Bits：校驗位 0 -- None (無校驗) 1 -- Odd (奇同位檢查)

限巨匠講師專用

## 藍牙 MeetAndroid Library 介紹

### 藍牙通訊協定/命令格式介紹

- Android to Arduino Data Format :



Event ID : A ~ Z or a ~ z

起始符號

ex : A1;-0.9876;1.7654 0x13  
c1 0x13

## 藍牙 MeetAndroid Library API 介紹

使用前需先宣告一個 MeetAndroid 物件，例如：

```
MeetAndroid AppsdüinoBT; //宣告一個 MeetAndroid 物件名稱叫 AppsdüinoBT
```

常用 API 如下：

- ◆ void registerFunction(void (\*)(uint8\_t, uint8\_t), uint8\_t) : 宣告接收到起始符號時的處理程序(processHandler)，例如：  
AppsdüinoBot.registerFunction(BTCommandHandler, 'c'); //指當接收到起始符號為 'c' 時跳至 BTCommandHandler() 程序處理(Call Back Function)
- ◆ bool receive(void) : 檢查是否 received buffer 有接收到資料，若有資料則回傳 true，並會自動跳至預設的處理程序執行，此函數需至於 loop() 內
- ◆ void getString(char[]) : 若收到的資料是字串形式(string)，則可以此函數來讀取
- ◆ int getInt() : 若收到的資料是整數，則可以此函數來讀取
- ◆ long getLong() : 若收到的資料是長整數，則可以此函數來讀取
- ◆ float getFloat() : 若收到的資料是浮點數(float)，則可以此函數來讀取
- ◆ void send(char) : 可用 send() 函數來傳送字元、字串、整數、長整數... 等不同資料
- ◆ void send(const char[]);
- ◆ void send(uint8\_t);
- ◆ void send(int);
- ◆ void send(unsigned int);
- ◆ void send(long);
- ◆ void send(unsigned long);
- ◆ void send(long, int);

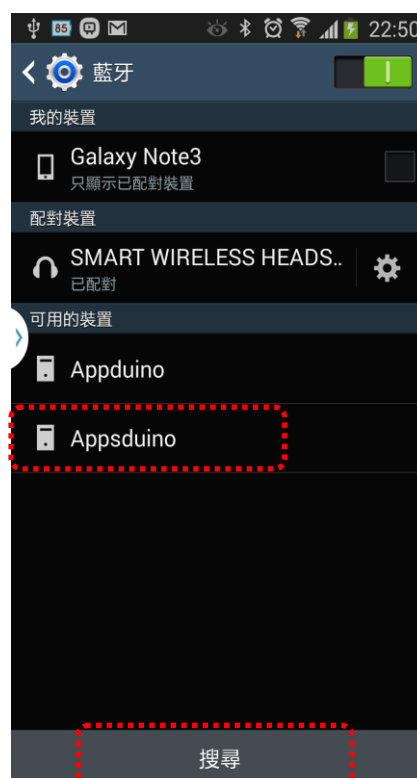
## 智慧家庭與 Android App 互動實驗

### 軟體安裝

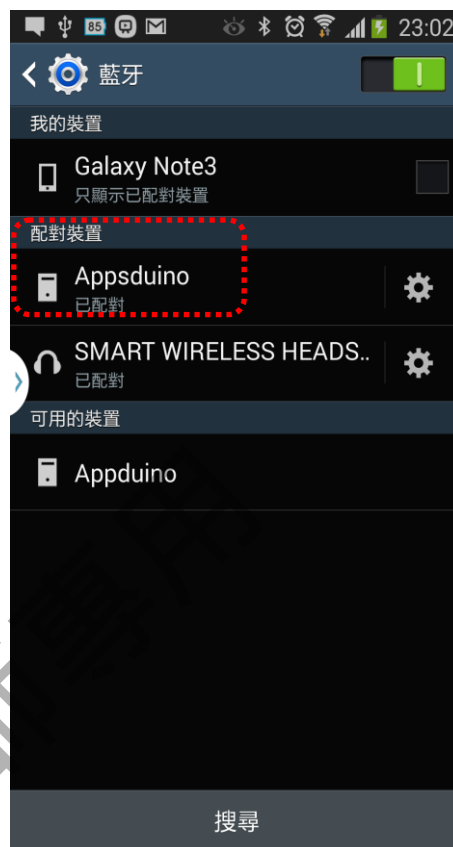
1. 首先需將光碟上 libraries 目錄內的程式庫拷貝到硬碟 Arduino IDE 所在的 libraries 目錄中(例如 C:\arduino\libraries 或 C:\Program Files(x86)\Arduino\libraries)，並離開 Arduino IDE 開發環境再重新進入。
2. 接著啟動 Arduino IDE，點選 File->open 開啟光碟上的程式(IoTSN\_RFID.ino)，將程式上傳到 Arduino 控制板內，上傳成功後系統自動重新啟動，LCD 上就會顯示感測傳來的溫溼度值及其他的數據(順序為溫度、濕度、照度、聲音、磁簧、兩個燈的開(1)與關(0)與亮度準位



3. 開啟藍牙功能，並且與 Appsdüino 藍牙模組完成配對  
先接上電源，同時會看到藍牙模組上的紅色小燈閃爍，代表等待配對連結。接著點選 Android 裝置的 [設定->藍牙->開啟藍牙功能->點選藍牙->搜尋裝置] 來完成藍牙設定。不同 Android 版本，操作畫面可能不太相同，但是都大同小異，請自行參考 Android 裝置的原廠說明書完成設定，以下以 Android 4.3 版本畫面為例說明，點選搜尋，過一會兒應該看到 Android 系統找到 Appsdüino 的藍牙裝置，請點選 Appsdüino



接著出現配對要求輸入密碼的畫面，請輸入密碼 0000，按下確定，就會看到配對成功，Appsdduino 藍牙模組已經加入到 Andorid 的已配對裝置清單之中



Note：可利用如下之” Config\_BT\_Name.ino” 程式更改藍牙 ID 以區別並易於辨識，只要更改紅色之名稱即可

```
//使用” AT+NAME” 命令更改藍牙的ID Name，綠燈亮表示更改完成
int LedGreen = 13 ;

void setup()
{
    pinMode(LedGreen, OUTPUT) ;
    digitalWrite(LedGreen, LOW) ; // turn off
    Serial.begin(57600); // Baud rate of HC-0x bluetooth module(設定已知之速率)
    delay(1000);
    Serial.write("AT+NAMEAppsdduino"); // Change the "Appsdduino" name to your ID name,
    改成你欲設定之名稱
    while(!Serial.available()) ; // waiting for 'ok' message
    Serial.read(); // flush feedback message
    digitalWrite(LedGreen, HIGH) ; // turn on LED
}

void loop()
{
}
```

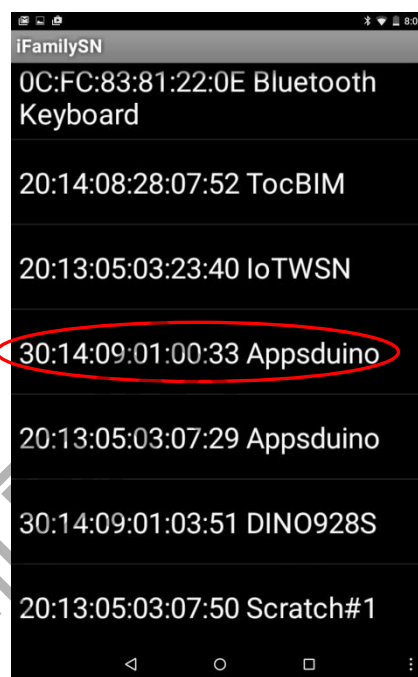
只要更改紅色框  
內之名稱即可



4. 請從 Google Play 下載 “IoTSN 互動軟體” 後安裝(如下頁圖示)或將光碟中的 iFamilySN.apk 軟體,複製到手機記憶體中,然後執行安裝到手持裝置內。執行 APP 並點選 [藍牙裝置]與選擇欲連線的藍牙設備(如下頁圖示),回到主畫面,看到選定的 Appsduino 藍牙模組的名稱,再點選按鍵 [連線] 完成連線



IoTSN Android App



選擇[藍牙裝置]與按下[連線]



稍後一會兒，螢幕出現 Device connected，同時藍牙模組上的紅燈停止閃爍，保持恆亮，就代表藍牙連線建立完成，可以互傳資料了。此時應該可以看到傳回來的溫、濕度、照度、聲音、磁簧、兩個燈的開/關狀態。

當把磁簧開關靠近，即顯示綠色“Close”，若磁簧開關移開，則顯示紅色“Open”，點選燈泡按鈕，則可控制繼電器動作，以便開或關燈。

◆ Android APP 聲控命令(請先將智慧型裝置的藍牙配對(PIN code:0000)並連上網路(WiFi/3G))

- “紅燈”：Led 紅燈 亮/滅
- “藍燈”：Led 藍燈 亮/滅
- “開燈”：E17 燈泡亮
- “關燈”：E17 燈泡滅



## 智慧家庭參考程式

### ◆ 主程式(IoTSN\_RFID.ino)

相關程式庫及副程式存放於光碟上，請一併複製到硬碟中即可編譯執行

```

/*
    Appsdüino IoT Project : IoTSN Sensor Network w/RFID & BT communication project : Tony Su 2015/04/30
*/

#include <Streaming.h>
/*-----( Import needed libraries )-----*/
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h> // F Malpartida's NewLiquidCrystal library

/*-----( Declare Constants )-----*/
#define I2C_ADDR    0x27 // Define I2C Address for the PCF8574T
//---(Following are the PCF8574 pin assignments to LCD connections )----
#define BACKLIGHT_PIN  3
#define LED_OFF  1
#define LED_ON   0
/*-----( Declare objects )-----*/
LiquidCrystal_I2C  lcd(I2C_ADDR,2,1,0,4,5,6,7); // declare I2C LCD object
/* -----  RC522 RFID Test program created by Tony Su 07/22/2013 ----
*
* Read a card using a mfrc522 reader on your SPI interface
* Pin layout should be as follows (on Arduino Uno):
* MOSI: Pin 11 / ICSP-4
* MISO: Pin 12 / ICSP-1
* SCK: Pin 13 / ICSP-3
* SS: Pin 10
* RST: Pin 9
*
*/

#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

```

```

RFID rfid(SS_PIN, RST_PIN);
//-----
#define Buzzer      8      // Buzzer Pin
#define WindowSW    7
#define RedLed      6
#define BlueLed     5
#define RelayIN1    4      // Define Relay Pin
#define RelayIN2    3
#define NULL        '\0'   // string null terminator
#define LightSensor A2
#define SoundSensor A3
//-----
#include <MeetAndroid.h> // include Android Bluetooth Library header
#include <Timer.h>
Timer tick ;// setup one sec timer event
//----- For HTU21D Temperature/Humddity measure -----
#include <HTU21D.h>
//Create an instance of the object
HTU21D myHumidity;
//-----
volatile int CdsThreshold = 300 ; // default cds threshold,turn on blue led if > CdsThreshold
volatile boolean CdsTriggerFlag = false ;
volatile int soundPeakValue = 0 ;// Check Sound Sensor and keep the peak value during one sec interval
//----- Melody -----
#include "pitches.h"
// notes in the melody:
int melody[] = {
    NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
    4, 8, 8, 4,4,4,4,4 };
//----- Android Bluetooth Test -----
MeetAndroid AppsdüinoBot;
//-----
String sensorMsg = "" ; // initial sensor Message string(include Temp,HR,Rly1/2,..)
//----- RFID information definition -----
String rfid_receive_msg = " Wait" ; // initialize rfid received message
String rfid_prereceive_msg = "x    " ; //
//----- Define Serial Number(4 bytes) and Name(7 chars maximum)

```

```

unsigned char RFIDserNum[][4] ={{0x33,0xCC,0xC3,0xD3},
                                {0xED,0x5A,0x63,0x7C},
                                {0xA5,0x6B,0x60,0xB6},
                                {0x35,0x22,0x10,0x02},
                                {0xE2,0x07,0xD3,0xCC},
                                {0x4C,0xC3,0x14,0x46},
                                {0x7C,0x93,0x14,0x46},
                                {0x8C,0xC3,0x14,0x46},
                                {0xAC,0x7D,0x14,0x46},
                                {0xD3,0x73,0xDD,0x80},
                                {0x19,0xD9,0x41,0x03},
                                {0xB5,0x07,0x33,0xB3}
                                };

//----- Maximun Name Length <= 7 chars to avoid too long App button Text -----
String  Name[] = { "Jenny"," Eddy"," John"," Mary",
                  "Nixon","James","Jasmi","Jenny",
                  "Stone","Wison","Steve"," Tony" };

//-----

void setup()
{
  //--- I/O definition -----
  pinMode(RedLed, OUTPUT);
  pinMode(BlueLed, OUTPUT);
  pinMode(RelayIN1, OUTPUT); // define Relay as Output Pin
  pinMode(RelayIN2, OUTPUT); //
  pinMode(LightSensor,INPUT_PULLUP); // enable pull up resister
  pinMode(WindowSW,INPUT_PULLUP); // enable pull up resister
  digitalWrite(RelayIN1,LOW ); // HIGH Active
  digitalWrite(RelayIN2,LOW );
  pinMode(Buzzer, OUTPUT);
  //----- RFID Initialize -----
  SPI.begin();
  rfid.init();
  //-----
  lcd.begin (16,2); // initialize the lcd
  // Switch on the backlight
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(LED_ON);
  lcd.backlight(); //Backlight ON if under program control

```

每片 RFID Tag 的 ID(4 Bytes)  
皆不同，使用前需先讀取內置  
的 ID，才能做後續對應的應用

```

lcd.setCursor(0,0); //Start at character 0 on line 0
// Configure your bluetooth module to this baud rate : 57600 bps
Serial.begin(57600);
// while(!Serial) ; // wait for serial port is ready
// register callback functions, which will be called when an associated event occurs.
// - the first parameter is the name of your function (see below)
// - match the second parameter ('A', 'B', 'a', etc...) with the flag on your Android application
AppsduinoBot.registerFunction(AppsduinoiFamilyIO, 's');
AppsduinoBot.registerFunction(SetCdsThreshold, 'p');
//----- initial HTD21D HR sensor -----
myHumidity.begin();// Initialize HTU21D Humidity Sensor
delay(2000) ; // waiting for 2 sec at least after power on for stable HR reading
tick.every(1000, updateSensorHandler);// read sensor data from I2C slave node
tick.every(10, HundredmsSoundHandler);// read temperature/humidity once per second
lcd.setCursor(0,0); //Start at character 0 on line 0
lcd.print("Service Now      ");
}

//----- Main Program -----
void loop()
{
    int  NameIndex ;

    //-----
    tick.update(); // check timer event
    AppsduinoBot.receive(); // you need to keep this in your loop() to receive events
    //----- Check RFID Name Card and show who is touching -----
    if (rfid.isCard()) {
        NameIndex = RFIDserNumLookup() ; // just show who is touching
        if(NameIndex != -1)
            rfid_receive_msg = Name[NameIndex] ;
        else
            rfid_receive_msg = "Other" ;
        TxName2CO(rfid_receive_msg) ;// Tx received msg to CO
    }
    rfid.halt(); // Enter Sleep Mode
    delay(50) ;
}

```



```
//-----
//--- 's' flag to control RGB Led and Melody Play ----
void AppsduinoiFamilyIO(byte flag, byte numOfValues)
{
    int randomvalue = AppsduinoBot.getInt();

    iFamilyIO(randomvalue); // execute the related function
}
//----- Toogle AC Light on/Off -----
//   Light # :
//           1 : Toogle Light #1 on/off
//           2 : Toogle Light #2 on/off
//-----

void iFamilyIO(int value)
{
    switch(value) {
        case 1 : // Toogle AC Light on/off
            digitalWrite(RelayIN1,!digitalRead(RelayIN1));
            break ;
        case 2 : // Tuon off AC Light
            digitalWrite(RelayIN2,!digitalRead(RelayIN2));
            break ;
        case 3 : // toggle Red led on /off
            digitalWrite(RedLed,!digitalRead(RedLed));
            break ;
        case 4 : // toggle Blue led on /off
            digitalWrite(BlueLed,!digitalRead(BlueLed));
            break ;
        case 5 : // melody play
            PlayMelody(); // play melody
            break ;
        case 6 : // Toogle AC Light on
            digitalWrite(RelayIN1,HIGH);
            break ;
        case 7 : // Toogle AC Light off
            digitalWrite(RelayIN1,LOW);
            break ;
    }
}
```

```
//----- Adjust CdsThreshold Level -----
//--- 'p' flag to setup CdsThreshold Level (value : 100,200,..,800) ----
void SetCdsThreshold(byte flag, byte numOfValues)
{
    CdsThreshold = AppsduinoBot.getInt();
}

//----- RFID Serial Number look up and matching -----
// Look up the RFIDserNum data base
// return : Name if matching
//          "None" if not found
//-----

int RFIDserNumLookup()
{
    int tagCount = sizeof( RFIDserNum)/4 ; // calculate the UID count

    if (rfid.readCardSerial()) {
        for(int i=0;i < tagCount;i++) {
            if (rfid.serNum[0] == RFIDserNum[i][0]
                && rfid.serNum[1] == RFIDserNum[i][1]
                && rfid.serNum[2] == RFIDserNum[i][2]
                && rfid.serNum[3] == RFIDserNum[i][3]
            )
                return i ;
        }
        return -1 ;
    }
}

/* Created 04/6/2014
   Transmit received NFC/RFID name to smart devices
*/
void TxName2CO(String datastr) //--
{
    lcd.setCursor(0, 1);      // Display from 2nd Row
    lcd << "RFID : " << datastr << "    ";

    if(rfid_prereceive_msg != datastr) {
        rfid_prereceive_msg = datastr ;
        Serial <<  datastr << 'z' ; // Tx RFID message to smart devices
    }
}
```

```
//----- Monitor Sound Sensor with 100ms interval & keep the peak value -----
//
//          Reset in the OneSecTimerHandler()
//-----
void HundredmsSoundHandler()
{
    int Soundvalue ;

    Soundvalue = analogRead(SoundSensor) ; // read sound sensor value
    if(Soundvalue > soundPeakValue)
        soundPeakValue = Soundvalue ;
}

//----- Read sensor data once 1 sec and Tx sensor data to Smart devices -----
// Data Format : Temp(xxx.x),Humidity(xx),Light1(x),Light2(x),LightSensor(xxx),Sound(xxx),DoorOpen(x),PIR
// Sensor(x) = 28 Bytes total include comma ","
// Timer Interval : 1 sec
//-----
void updateSensorHandler() //-- update per sec --
{

    int cdsvalue ;
    String sensorMsg = "" ; // Initial the Sensor message to transmit it to Master with BT

    sensorMsg += (int) (myHumidity.readTemperature() * 10) ;// temperature for test
    char lastDigit = sensorMsg.charAt(sensorMsg.length()-1) ;
    sensorMsg.setCharAt(sensorMsg.length()-1, '.') ; // divide 10.0 to get real temperature
    sensorMsg += lastDigit ; // Add '.' separator
    sensorMsg += ',' ; // Add ',' separator
    sensorMsg += (int) myHumidity.readHumidity() ;
    sensorMsg += ',' ; // insert comma "," to be serarate delimiter

    //----- Check CDS Level if need to turn on Led -----
    cdsvalue = analogRead(LightSensor) ; // read CDS current value
    if(cdsvalue > CdsThreshold) {
        digitalWrite(BlueLed,HIGH) ; // turn on blue Led
        CdsTriggerFlag = true ; } // set CdsTriggerFlag = true
    else if(CdsTriggerFlag) {
        digitalWrite(BlueLed,LOW) ; // turn off blue Led
        CdsTriggerFlag = false ; }
    sensorMsg += cdsvalue ; // read CDS current value

    //-----
    sensorMsg += ',' ; // insert comma "," to be serarate delimiter
```

```

sensorMsg += soundPeakValue ; // Insert Sound peak Value
soundPeakValue = 0 ; // reset the sound peak value
sensorMsg += ',' ; // insert comma "," to be serarate delimiter
sensorMsg += digitalRead(WindowSW) ; // read co current value
sensorMsg += ',' ; // insert comma "," to be serarate delimiter
sensorMsg += digitalRead(RelayIN1) ; // convert to ASCII(x : 0/1)
sensorMsg += ',' ; // insert comma "," to be serarate delimiter
sensorMsg += digitalRead(RelayIN2) ; // convert to ASCII(x : 0/1)
sensorMsg += ',' ; // insert comma "," to be serarate delimiter
sensorMsg += CdsThreshold ; // Tx CdsThreshold Level

//----- Display current sensor data in the 1602 LCD display -----
lcd.setCursor(0,0);      // Display from 1st Row
int comaPos = sensorMsg.indexOf(',');
comaPos = sensorMsg.indexOf(',',comaPos+1);
comaPos = sensorMsg.indexOf(',',comaPos+1); // Total look for third ','
comaPos = sensorMsg.indexOf(',',comaPos+1); // Total look for forth ','
lcd << sensorMsg.substring(0, comaPos+1) << "  ";
lcd.setCursor(0,1);      // Display from 2nd Row
lcd << sensorMsg.substring(comaPos+1, sensorMsg.length()) << "  ";
Serial <<  sensorMsg << 'z' ;
}

//----- Play melody from Arduino examples toneMelody ----
void PlayMelody() {
    // iterate over the notes of the melody:
    for (int thisNote = 0; thisNote < 8; thisNote++) {

        // to calculate the note duration, take one second
        // divided by the note type.
        //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
        int noteDuration = 1000/noteDurations[thisNote];
        tone(8, melody[thisNote],noteDuration);

        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        // stop the tone playing:
        noTone(8);
    }
}

```